

Robust Additive Randomized Encodings With Transparent Setup

Sapir Bekerman Freizeit
Tel Aviv University

Nir Bitansky
New York University

September 8, 2026

Abstract

Additive randomized encodings (AREs) reduce the computation of an m -party function $f(x_1, \dots, x_m)$ to independently encoding each input and adding the resulting encodings in an Abelian group, so that their sum reveals nothing but the output. Robust AREs (RAREs) additionally protect against collusions between the evaluator and some of the parties, guaranteeing that even if some of the input encodings are completely controlled by the adversary, the partial sum of the remaining (honest) input encodings leaks no more than the residual function $f(\mathbf{x}_H, \cdot)$ obtained by fixing the honest inputs $\mathbf{x}_H$. Much of the appeal of RAREs come from the fact that they imply *robust non-interactive secure computation* in the *shuffle model*, where parties send messages to the evaluator through anonymous communication channels.

Existing constructions of general-purpose RARE either assume *ideal obfuscation* or rely on structured public parameters generated using secret randomness that must remain hidden. We construct general-purpose RARE with *transparent setup*. Assuming indistinguishability obfuscation (which is necessary) and LWE, both with subexponential security, we show that every efficient function admits an efficient RARE in the common random string model. We also show how to eliminate public parameters altogether in the honest-majority setting. The main technical observation behind our construction is that *non-robust* ARE, which exist assuming only one-way functions, can be leveraged to protect from *spoofing* of honest party encodings.

Contents

1	Introduction	2
1.1	Our Results	3
1.2	Technical Overview	4
1.3	More Related Work	7
1.4	Open Questions	8
2	Preliminaries	9
2.1	Robust Additive Randomized Encodings	9
2.2	Multi-String Tagged NIZK	11
2.2.1	Instantiation	13
2.2.2	Single-String Tagged NIZK	14
2.3	Multi-Key Homomorphic Encryption	15
2.4	Puncturable Pseudorandom Functions	17
2.5	Indistinguishability Obfuscation	18
2.5.1	Nondeterministic Probabilistic iO	19

2.6	Programmable Signatures	19
2.6.1	Instantiation	20
2.7	Non-Interactive Commitments	21
3	RARE in the Common Random String Model	21
3.1	Construction	21
3.2	Security Analysis	26
4	Setup-Free RARE for an Honest Majority	29
4.1	Construction	30
4.2	Security Analysis	37
A	From Neighboring-Input $\mathcal{H}$-Robustness to Full $\mathcal{H}$-Robustness	51

1 Introduction

Introduced by Halevi, Ishai, Kushilevitz, and Rabin [HIKR23], *additive randomized encodings* (AREs), reduce secure computation to its simplest nontrivial operation: addition. To compute an m -party function $f(x_1, \dots, x_m)$, every party i independently maps its input x_i to a randomized encoding $\hat{x}_i$ in an Abelian group. Then, an evaluator given the sum $\hat{y} = \sum_{i \in [m]} \hat{x}_i$ can decode the result $y = f(x_1, \dots, x_m)$, but learns nothing else regarding the parties' private inputs.

While the vanilla notion of ARE only offers security against the evaluator, the stronger notion of *robust* ARE (RARE) also withstands collusion between some of the parties and the evaluator. In this setting, the best we can hope for is that the partial sum $\hat{y}_H := \sum_{i \in H} \hat{x}_i$, corresponding to the inputs $\mathbf{x}_H$ of honest parties $H \subseteq [m]$, does not reveal more than the residual function

$$f_{H, \mathbf{x}_H}(\cdot) := f(\mathbf{x}_H, \cdot),$$

defined over the inputs of the corrupted parties $\overline{H}$. Indeed, the residual function inevitably leaks as the adversary has complete control over the encodings $\hat{x}_{\overline{H}}$.

Much of the appeal of AREs lies in the direct route they offer to *non-interactive secure computation*, a longstanding goal pursued across a variety of models such as correlated private randomness or a public-key infrastructure [FKN94, HLP11, BGI+14, GGG+14, HIJ+16, HIJ+17]. AREs imply non-interactive secure computation assuming only *anonymous communication*. This is formally captured by the fact that addition can be robustly realized in the *shuffle model* [IKOS06], where parties can anonymously send multiple messages to an evaluator, and is known as *shuffle MPC*.

The Feasibility of AREs and RAREs. Following a sequence of works [HIKR23, BEG25, BEGI26] our understanding of the feasibility of ARE is by now rather complete. Any function admits an information-theoretic ARE, and, assuming one-way functions, efficient functions admit efficient AREs. The situation for RARE is far less understood, both information theoretically and computationally.

In this work, we focus on the computational setting and seek efficient RAREs for general efficiently computable functions. Here there is an inherent obfuscation barrier. Even for two parties, general-purpose RARE implies program obfuscation [HIKR23]. In particular, one cannot hope for a general simulation-based definition in which the partial sum encoding $\hat{y} = \sum_{i \in H} \hat{x}_i$ can be efficiently simulated from black-box access to the residual function: this would amount to virtual-black-box obfuscation, which is generally impossible [BGI+01]. Instead, the aim is indistinguishability: if

two sets of honest inputs $\mathbf{x}_H^0, \mathbf{x}_H^1$ induce identical residual functions $f_{H, \mathbf{x}_H^0} \equiv f_{H, \mathbf{x}_H^1}$, then the corresponding partial sums $\hat{y}_H^0, \hat{y}_H^1$ should be computationally indistinguishable. This is analogous to the guarantee of indistinguishability obfuscation (iO), which is indeed necessary for general-purpose RARE.

Halevi et al. [HIKR23] give a proof-of-concept construction by combining ideal obfuscation with ideal resettable MPC. This provides a useful conceptual blueprint, but is not a realizable construction: the corresponding ideal primitives are generally impossible. Bitansky and Freizeit [BF24] subsequently show a RARE for general functions based on iO and on a new primitive called pseudo-non-linear codes, which they construct from either LWE or DDH.

Setup and Transparency. The main caveat of the construction of Bitansky and Freizeit [BF24] is that it relies on structured public parameters. The public parameters need to be generated in a private trusted setup phase. Security crucially relies on the setup randomness never leaking, once leaked the construction completely breaks down. Borrowing terminology from the literature on succinct proof systems [BBHR18], we seek a *transparent setup*: any common parameters should be generated entirely from public randomness, without secret coins, trusted erasure, or “toxic waste”. Such transparency is the gold standard for cryptographic public parameters. We therefore ask:

*Can robust additive randomized encodings be realized with a transparent setup?
or perhaps even with no setup whatsoever?*

1.1 Our Results

Our first result is a construction of general purpose RARE in the *common random string model*.

Theorem 1. *Assuming iO and LWE, both with subexponential security, any efficient function has an efficient RARE in the common random string model.*

Our second result does not rely on any common randomness, but is restricted to the setting of an *honest majority*; namely, it assumes that at most $\lfloor (m-1)/2 \rfloor$ parties collude with the evaluator.

Theorem 2. *Assuming iO and LWE, both with subexponential security, any efficient function has a setup-free efficient RARE assuming an honest majority.*

On the Assumptions. As already noted, iO is a necessary assumption for constructing RARE. Constructions of iO are by now known from well-founded (subexponential) assumptions [JLS21, JLS22]. Beyond iO, we use LWE to obtain *multi-key homomorphic encryption* (MHE) [CM15] with appropriate partial-decryption simulatability [MW16], which is not currently known to follow from iO alone.¹ By instantiating the MHE using the spooky encryption scheme of [DHRW16], we can replace LWE with DDH, albeit only for functions in NC^1 .

Additional Features. Beyond transparency, an additional advantage of our CRS-construction over [BF24] is the size of the CRS. In [BF24] the public parameters grow with the circuit size of the computed function. In our CRS-solution the CRS grows only with the circuit depth of the function and the (total) size of inputs, but not with its circuit size. In fact, the growth in these parameters is an artifact of the LWE-based MHE construction and does not seem inherent. A stronger MHE, with no CRS, and a stronger partial decryption simulatability would lead to a CRS that only depends on the security parameter. In fact, assuming DDH, such a scheme is already shown in [DHRW16],

¹Conceivably, such MHE could be constructed from iO and one-way functions: we do not need compactness and are therefore not subject to the known iO barriers [AS15, BDV17, BD19].

but only for NC^{1,2}. As in [BF24], our encodings $\widehat{x}_i$ can be decomposed to a non-interactive part $\widehat{z}_i$ sent directly to the evaluator, without any trusted summation (or anonymous communication), and a group part $\widehat{g}_i$ that requires the extra resource of summation (or anonymous communication). Our group encodings are *succinct* in the sense that they do not scale with the complexity of the underlying function.

1.2 Technical Overview

We start by reviewing the basic difficulties in constructing RARE and, more broadly, non-interactive MPC, and by placing our construction in the context of existing work.

Resetting and Spoofing. In the presence of corrupted parties, non-interactive secure computation faces two inherent difficulties [HLP11, HJ+17]. The first is the possibility of so-called *resetting attacks*: corrupted parties can repeatedly resample their own messages and, in particular, select different inputs for themselves. Accordingly, as discussed above, we aim only for *best-possible security*: corrupted parties should not be able to distinguish two executions in which the two collections of honest inputs $\mathbf{x}_H^0$ and $\mathbf{x}_H^1$ induce the same residual function,

$$f(\mathbf{x}_H^0, \cdot) \equiv f(\mathbf{x}_H^1, \cdot).$$

The second inherent difficulty is that corrupted parties may try to *spoof* the messages of some honest parties in an attempt to learn about the inputs of other honest parties. For instance, consider a three-party setting in which the first two parties are honest and the third is corrupted. If there exist inputs x_1^0, x_1^1, x_2, x_2^* such that

$$f(x_1^0, x_2, \cdot) \equiv f(x_1^1, x_2, \cdot), \quad \text{but} \quad f(x_1^0, x_2^*, \cdot) \neq f(x_1^1, x_2^*, \cdot),$$

then an adversary that manages to replace the honest input x_2 with x_2^* learns information about the first honest input and can distinguish x_1^0 from x_1^1 . Preventing such spoofing attacks inherently requires an additional resource that essentially *glues the honest messages together*. In the context of RARE, this additional resource is the summation oracle.

The HIKR Blueprint and the BF Construction. In light of the above, Halevi, Ishai, Kushilevitz, and Rabin [HIKR23] suggest a general blueprint for constructing RARE: first, take a *resettable MPC protocol* [GS09, GM11], which retains security even when the adversary can reset the honest parties and rechoose their messages; then obfuscate *the next-message function* of each party to make the protocol non-interactive; and finally use the summation oracle to create a mechanism for gluing the honest inputs together. They show how to instantiate this blueprint assuming *ideal obfuscation* and *ideal resettable MPC*, resulting in an *ideal RARE*. As mentioned before, this instantiation is viewed as a “proof of concept”, since both the ideal primitives from which they start and the resulting ideal RARE are subject to impossibility results stemming from the impossibility of ideal obfuscation [BGI+01]. The technical challenge is therefore to replace the ideal, simulation-based obfuscation and resettable MPC with non-ideal, indistinguishability-based counterparts, with the goal of obtaining a non-ideal, indistinguishability-based RARE.

Toward this goal, Bitansky and Freizeit [BF24] partially diverge from the HIKR blueprint. At a high level, using indistinguishability obfuscation (iO), they show how to obfuscate a trusted party that receives authenticated input encodings from the parties, verifies them, and performs the entire

²The authors conjecture that this can be extended to all efficient functions, but this has yet to be established [DHRW16, Remark 4.5].

computation. The authentication mechanism guarantees the gluing of honest inputs and is realized using the summation oracle and a new primitive called *pseudo-nonlinear codes*. The downside of their approach is that the resulting RARE has *structured and non-transparent public parameters*. In particular, the obfuscated trusted party and the corresponding public encoding keys must be privately generated and then published to all parties. If the randomness used to generate these parameters is leaked all bets are off.

Construction I: Making the Setup Transparent. To achieve a transparent setup, we return to the HIKR blueprint and aim to identify a resettable MPC and a summation-based gluing mechanism that are both *iO-friendly* and compatible with each other. For the iO-friendly resettable MPC component, we turn to the work of Halevi et al. [HIJ+17], who construct non-interactive secure computation assuming a public-key infrastructure (PKI) and a common random string. To this end, they make the MPC protocol of Mukherjee and Wichs [MW16] resettable using iO. They then use the PKI setup to glue the honest inputs together.

Recall that in a PKI setup, every party has its own secret and public keys, and each party's public key is registered with all other parties. In the construction of Halevi et al. [HIJ+17], these are signing and verification key pairs $(sk_1, vk_1), \dots, (sk_m, vk_m)$. Each party i signs its messages, which encode its input, using sk_i . The obfuscated next-message function of party i first uses the registered keys $vk_1, \dots, vk_{i-1}, vk_{i+1}, \dots, vk_m$ of the other parties to verify their signatures and only then produces its output. This ensures that, as long as a party is honest and its secret signing key remains hidden, its messages and input cannot be spoofed.

A New Enabler: Non-Robust 2ARE. Our main observation is that we can replace the above PKI-based authentication mechanism with a summation-based mechanism. In fact, for this purpose we can use any *non-robust* two-party ARE (2ARE). A 2ARE for a function $g(x, y)$ guarantees that, if neither party is corrupted, the corresponding sum encoding $\widehat{g}(x, y)$ reveals $z = g(x, y)$ to the evaluator and nothing more; it can be efficiently simulated from z . In contrast, if either party is corrupted, there is no security whatsoever for the other party: the adversary learns all private inputs. Such a 2ARE is now known to exist for every efficiently computable function g , assuming only one-way functions [BEG12].

Our basic idea is to use a 2ARE for the function $g(sk, m)$ that, given a signature key sk from one party, signs the message m provided by the other party. Here we cannot rely on the same authentication pattern as in the PKI case. Indeed, whenever a party uses a signing key sk in a 2ARE with a corrupted party, sk leaks. Instead, we use a pairwise authentication pattern. Each party i samples $m - 1$ key pairs $(sk_{i,j}, vk_{i,j})$, one for every other party $j \neq i$. It then runs a 2ARE for $g(sk_{i,j}, m_{j,i})$ with each such party j , using $sk_{i,j}$ to sign a message $m_{j,i}$ of party j . Roughly speaking, $m_{j,i}$ is a message sent from j to i in the underlying protocol. Party i 's obfuscation has all the verification keys $vk_{i,j}$ hardwired and verifies the signatures before producing its output.

For every honest parties i, j , the keys $sk_{i,j}, sk_{j,i}$ remain hidden, and hence messages $m_{j,i}, m_{i,j}$ between honest parties cannot be spoofed. For a corrupted party j^* , however, sk_{i,j^*} is leaked. The adversary can therefore execute party i 's obfuscation with different messages $m_{j^*,i}$; indeed, j^* can sign any message to be verified under vk_{i,j^*} . However, this is already covered by the resettable security of the underlying MPC protocol. Leakage also occurs in the reverse direction: the adversary learns m_{i,j^*} , the message from i to j^* , but this, too, is covered by the underlying protocol.

The Concrete Protocol. To make the above intuition more concrete, in the resettable protocol of Halevi et al. [HIJ+17], the messages $m_{j,i}$ sent by j to the other parties are all intended to be the same message, $m_j = \text{Enc}_{sk_j}(x_j)$, consisting of a multi-key homomorphic encryption (MHE) of its input x_j . The next-message function EV_j of party j receives input encryptions from all parties,

homomorphically evaluates f , and uses the secret key sk_j to produce a *partial-decryption* value p_j . The values $p_1, \dots, p_m$ can then be combined by the evaluator to recover the output of the function. The 2ARE authentication mechanism essentially fixes the encryptions $\text{Enc}_{sk_j}(x_j)$ for every honest j , thereby fixing the honest inputs $\mathbf{x}_H$. In contrast, the encryptions $\text{Enc}_{sk_{j^*}}(x_{j^*})$ of corrupted parties j^* are under the full control of the adversary, who can accordingly homomorphically evaluate $f(\mathbf{x}_H, \cdot)$ on any choice of corrupted inputs $\mathbf{x}_{\overline{H}}$. Provided that the underlying MHE satisfies a standard decryption-simulation property, this does not leak anything beyond the residual function, in an indistinguishability-based sense.

While the idea of using 2ARE for pairwise authentication is quite general, the above transformation cannot be combined with resettable MPC in a fully generic way, and we essentially need to prove its security from scratch. Indeed, the notion of an iO-friendly resettable MPC is not formal. As usual, using iO introduces various technical difficulties and requires iO-compatible primitives. For our (simpler) construction in the common random string model, we rely mostly on techniques similar to those of Halevi et al. [HIJ+17], which are now standard in the context of iO.

We refer to Section 3 for details.

Where Is the Common Random String Used? So far, we have not explicitly mentioned any use of a common random string. Such a string is, however, used as part of the underlying resettable MPC. Specifically, the security proof must establish that the adversary-controlled encryptions $\text{Enc}_{sk_{j^*}}(x_{j^*})$ fed to an honest party's obfuscated next-message function are well formed. Furthermore, we need to be able to efficiently extract the corresponding secret key sk_{j^*} . Both are required to invoke decryption simulatability when arguing that repeated homomorphic evaluations on adversarially chosen inputs $\mathbf{x}_{\overline{H}}$ do not reveal more than the residual function $f(\mathbf{x}_H, \cdot)$.

This requires a non-interactive zero-knowledge proof (NIZK) with an appropriate *simulation-extractability* property [Sah99, Gro06], which in turn is realized in the common random string model. Specifically, using standard techniques, we construct a *tagged NIZK* in which the common random string can be reprogrammed to a pseudorandom string corresponding to a designated party j . For the programmed string, proofs from party j are zero knowledge (can be simulated without the witness), whereas proofs from any other party $i \neq j$ allow extraction of a witness (and in particular are sound).

We in fact also use the common random string to support the multi-key homomorphic encryption scheme. However, this seems less inherent. In particular, if we restrict attention to NC^1 functions, a CRS-free MHE scheme is known from iO and DDH [DHRW16]. We return to this point below.

Construction II: Trading Common Randomness for an Honest Majority. We now discuss our second construction, which has no setup whatsoever but is secure only under an honest-majority assumption. The basic idea is to replace the NIZK in the CRS model with a multi-string NIZK (MNIZK) [GO07]. In an MNIZK, instead of relying on a single trusted CRS ρ , m parties independently generate strings $\rho_1, \dots, \rho_m$. Proving and verification are performed with respect to the aggregate string $\boldsymbol{\rho} = (\rho_1, \dots, \rho_m)$. The corresponding system is secure provided that a majority of the strings were generated honestly. For our purposes, we define a tagged variant with the following property: given an honest majority $H \subseteq [m]$, we can generate simulated strings $\tilde{\boldsymbol{\rho}}_H$ that are indistinguishable from the corresponding real strings $\boldsymbol{\rho}_H$. For every malicious choice of the other strings $\boldsymbol{\rho}_{\overline{H}}$, proofs from any party $j \in H$ are zero knowledge (can be simulated using a trapdoor) whereas proofs from any $j \notin H$ are extractable (using a trapdoor).³

Still, we cannot simply replace the CRS-NIZK with an MNIZK. In particular, the parties

³In fact, for our purposes, a simpler requirement focusing on a single party suffices, but in this overview the reader may think of the above intuitive requirement.

cannot compute proofs during the encoding phase, since each party has access only to their own MNIZK string. However, we can view proof generation as an extra round in the resettable MPC protocol and collapse this round using obfuscation. Namely, each party j generates an obfuscation of a program P_j that, given the other strings ρ_{-j} , generates an MNIZK proof that its message is well formed. As in the previous construction, the parties use 2ARE to authenticate the strings ρ , and the program P_j checks the corresponding signatures. Making this work requires care. For example, switching the obfuscated proof generator P_j to an obfuscated proof simulator requires the zero-knowledge indistinguishability error to vanish exponentially in the length of the multi-string ρ . We show how to construct suitable MNIZKs assuming subexponential iO, non-interactive witness indistinguishability (and one-way functions).

On Multi-Key Homomorphic Encryption and Common Randomness. As mentioned above, a main component of both constructions is multi-key homomorphic encryption (MHE) satisfying a *partial-decryption simulation* property. An MHE scheme for general functions with the required simulation property is known from LWE in the common random string model [MW16]. Alternatively, such a scheme can be obtained in the plain model assuming iO and DDH, but so far only for functions in NC^1 [DHRW16]. Other constructions (cf. [AJJM20]) satisfy a weaker simulation property that does not seem to suffice.

In Construction I, the MHE CRS η can be included in the overall CRS and therefore presents no problem. In Construction II, however, this is an issue. We show how to solve it at the cost of making the construction more complicated. The basic idea is to generate the MHE CRS η in a distributed manner from m independently generated strings $\eta_1, \dots, \eta_m$. Simply XORing these strings does not work, because the adversary can choose its strings $\eta_{\overline{H}}$ adaptively based on the strings η_H of the honest parties.

We solve this using *reverse randomization* [Lau83, Nao91]. We let the parties choose their strings η_j using a sufficiently sparse pseudorandom generator and use the MNIZK to prove that they did so. We then XOR the resulting strings η_j . While malicious parties are forced to follow this procedure in order to pass MNIZK verification, the honest parties can choose their strings to be truly random and simulate the MNIZK proofs. This guarantees that the adaptive choice of $\eta_{\overline{H}}$ is restricted to a tiny set that, when shifted at random, avoids *bad* MHE strings η with overwhelming probability.

This solution introduces additional complications. In particular, MHE encryptions can no longer be generated during the encoding phase, since the MHE CRS has not yet been determined. This adds another round to the underlying resettable protocol: (1) send MHE strings η and MNIZK strings ρ ; (2) generate MNIZK proofs that the strings η are pseudorandom; (3) generate MHE encryptions of the inputs together with MNIZK proofs of well-formedness; and (4) perform MHE evaluation and partial decryption. This results in three corresponding obfuscated programs. One side effect is that we can no longer authenticate input encryptions during encoding, since the inputs are encrypted only later. Instead, we authenticate commitments to the inputs and later prove their consistency with the generated encryptions. Other difficulties arise from stacking obfuscated programs and from the need to carefully choose the hardness of each primitive relative to the others.

We refer the reader to Section 4 for more details.

1.3 More Related Work

Randomized Encodings and Multiparty Randomized Encodings. Randomized encodings (REs), introduced by Applebaum, Ishai, and Kushilevitz [IK00, AIK06], reduce the computation of a function f to that of a simpler randomized function while revealing only the output of f .

Their extension to multiparty randomized encodings (MPREs), by Applebaum, Brakerski, and Tsabari [ABT18], reduces the computation of a multiparty function f to local preprocessing by each party followed by a simple joint computation g , such as a low-degree function, over the locally processed inputs. AREs can be viewed as a restricted form of MPREs in which the joint computation g is addition over an Abelian group; the robust notion additionally protects against coalitions of input parties and the evaluator [HIKR23]. Here we can only hope for residual function security, whereas when the degree of g is higher, full security is feasible.

Additive and Robust Additive Randomized Encodings. Halevi, Ishai, Kushilevitz, and Rabin introduced ARE and RARE, gave a general computational construction of ARE from a bilinear-group assumption, and obtained general RARE in an ideal-obfuscation model [HIKR23]. Bitansky, Erabelli, and Garg constructed ARE from any public-key encryption scheme [BEG25]. Bitansky, Erabelli, Garg, and Ishai showed that every function admits a statistically secure ARE; their construction is efficient for NC^1 and NL , and yields computationally secure AREs for all efficient functions assuming also one-way functions (or more generally assuming garbling schemes) [BEG126].

Bitansky and Freizeit gave a plain-model construction of general RARE (with non-transparent public parameters), based on iO and pseudo-non-linear codes constructed from either LWE or DDH [BF24]. Wei, Singh, Agrawal, and Zikas construct balanced AREs, which distribute the computation evenly among the parties, and used them to obtain client-efficient protocols for computational differential privacy in the shuffle model [WSAZ26].

Other Approaches to Non-Interactive MPC. Feige, Kilian, and Naor introduced the private simultaneous messages (PSM) model, in which parties share correlated *private* randomness and each sends one message to the evaluator [FKN94]. Halevi, Lindell, and Pinkas considered a web-based client-server model in which clients need not be online simultaneously, but may interact sequentially with the server [HLP11].

Beimel et al. formally introduced NI-MPC with correlated randomness and its robust, residual-function-based security notion, obtaining general computational feasibility from multi-input functional encryption and information-theoretic results for restricted functions [BGI+14]. Halevi et al. placed these models in a general framework of restricted interaction patterns, proved the completeness of the star topology, and showed that correlated randomness is necessary for general functions in that framework [HIJ+16]. Halevi et al. subsequently obtained NI-MPC against arbitrary collusions from a PKI and a CRS, using subexponentially secure iO and DDH [HIJ+17].

The shuffle model and shuffle MPC were introduced by Ishai, Kushilevitz, Ostrovsky, and Sahai [IKOS06]. They showed that anonymous communication over insecure channels suffices to implement unconditionally secure point-to-point channels, and hence general MPC in the honest-majority setting. They also show how to realize robust addition in the shuffle model.

1.4 Open Questions

Our work leaves several questions open.

1. Can setup-free RARE be constructed in the dishonest-majority setting?
2. Can the LWE and DDH assumptions be eliminated, yielding a construction based only on iO and one-way functions?
3. In a different vein, which functions admit information-theoretic RARE, possibly without any efficiency requirement?

2 Preliminaries

In this section, we define the primitives that we rely on.

Conventions and Notation. Throughout the paper, $\lambda \in \mathbb{N}$ denotes the security parameter, and all polynomial bounds are in λ . We fix polynomially bounded functions $n = n(\lambda)$ and $m = m(\lambda)$, denoting the length of each party's input and the number of parties, respectively. Thus, unless stated otherwise, an m -party input is a vector $\mathbf{x} = (x_1, \dots, x_m) \in (\{0, 1\}^n)^m$.

Honest algorithms are uniform: randomized algorithms are probabilistic polynomial time (PPT), and deterministic algorithms are deterministic polynomial time (DPT). Unless stated otherwise, adversaries and distinguishers are nonuniform families of polynomial-size Boolean circuits.

A function $\mu: \mathbb{N} \rightarrow \mathbb{R}_{\geq 0}$ is *negligible* if for every polynomial p it holds that $\mu(\lambda) < 1/p(\lambda)$ for every large enough λ . An event occurs with *overwhelming probability* if its probability is at least $1 - \mu(\lambda)$ for some negligible function μ .

For distribution ensembles $X = \{X_\lambda\}_{\lambda \in \mathbb{N}}$ and $Y = \{Y_\lambda\}_{\lambda \in \mathbb{N}}$, we write $X \approx_c Y$ if for every polynomial-size circuit family $D = \{D_\lambda\}_{\lambda \in \mathbb{N}}$,

$$|\Pr[D_\lambda(X_\lambda) = 1] - \Pr[D_\lambda(Y_\lambda) = 1]| \leq \text{negl}(\lambda).$$

We write $X \approx_s Y$ if their statistical distance is negligible, and $X \equiv Y$ if they are identically distributed. All probabilities are over the randomness of the sampled variables and algorithms involved. We write $x \leftarrow X$ for sampling x from a distribution X , and $y \leftarrow A(x)$ for running a randomized algorithm A ; we write $y \in A(x)$ to denote that y is in the support of $A(x)$. For deterministic assignment we write $y := A(x)$. For $q \in \mathbb{N}$, let $[q] := \{1, \dots, q\}$, and let $\parallel$ denote string concatenation. Vectors are denoted by bold letters. For $\mathbf{a} = (a_1, \dots, a_m)$ and $H \subseteq [m]$, let $\overline{H} := [m] \setminus H$, $\mathbf{a}_H := (a_i)_{i \in H}$ and $\mathbf{a}_{-i} := (a_j)_{j \in [m] \setminus \{i\}}$, with entries ordered by their indices. For a program P , we write $P[X](Y)$ to indicate that X is hardwired into the program and Y is supplied as its input.

2.1 Robust Additive Randomized Encodings

Additive randomized encodings and their robust variant were introduced by Halevi, Ishai, Kushilevitz, and Rabin [HIKR23]. Following Bitansky and Freizeit [BF24], we use robust additive randomized encodings in which each local encoding consists of a non-interactive part and a group element. Only the group elements must be added together by the summation oracle.

Let $f: (\{0, 1\}^n)^m \rightarrow \{0, 1\}^*$ be an efficiently computable m -party function.

Definition 1 (Robust additive randomized encoding). *A robust additive randomized encoding (RARE) for f over an Abelian group $(G, +)$ consists of PPT algorithms*

$$\text{RARE} := (\text{Setup}, \text{Enc}, \text{Dec}).$$

The setup algorithm generates public parameters

$$pp \leftarrow \text{Setup}(1^\lambda),$$

Given an input $x_i \in \{0, 1\}^n$ and an index $i \in [m]$, the encoding algorithm outputs

$$(\widehat{z}_i, \widehat{g}_i) \leftarrow \text{Enc}_{pp}(x_i, i), \quad \widehat{z}_i \in \{0, 1\}^*, \quad \widehat{g}_i \in G.$$

Given the vector $\widehat{\mathbf{z}} := (\widehat{z}_1, \dots, \widehat{z}_m)$ and an element $\widehat{g} \in G$, the decoder outputs

$$y \leftarrow \text{Dec}_{pp}(\widehat{\mathbf{z}}, \widehat{g}).$$

The scheme satisfies the following properties.

1. **Perfect correctness.** For every $\mathbf{x} := (x_1, \dots, x_m) \in (\{0, 1\}^n)^m$, if

$$\begin{aligned} pp \leftarrow \text{Setup}(1^\lambda), \quad (\widehat{z}_i, \widehat{g}_i) \leftarrow \text{Enc}_{pp}(x_i, i) \quad \text{for } i \in [m], \\ \widehat{\mathbf{z}} := (\widehat{z}_1, \dots, \widehat{z}_m). \end{aligned}$$

then

$$\Pr \left[\text{Dec}_{pp} \left(\widehat{\mathbf{z}}, \sum_{i \in [m]} \widehat{g}_i \right) = f(\mathbf{x}) \right] = 1.$$

2. **$\mathcal{H}$ -robustness.** Let $\mathcal{H} \subseteq 2^{[m]}$ be a family of subsets of parties. For $H \in \mathcal{H}$ and inputs $\mathbf{x}_H = (x_i)_{i \in H}$, define the residual function

$$f_{H, \mathbf{x}_H}(\mathbf{y}_{\overline{H}}) := f(\mathbf{x}_H, \mathbf{y}_{\overline{H}}), \quad \overline{H} := [m] \setminus H.$$

The scheme satisfies $\mathcal{H}$ -robustness if, for every $H \in \mathcal{H}$ and every $\mathbf{x}_H^0, \mathbf{x}_H^1$ such that $f_{H, \mathbf{x}_H^0} \equiv f_{H, \mathbf{x}_H^1}$, the distributions

$$\left(pp, \widehat{\mathbf{z}}_H^0, \sum_{i \in H} \widehat{g}_i^0 \right) \quad \text{and} \quad \left(pp, \widehat{\mathbf{z}}_H^1, \sum_{i \in H} \widehat{g}_i^1 \right)$$

are computationally indistinguishable, where

$$pp \leftarrow \text{Setup}(1^\lambda), \quad (\widehat{z}_i^b, \widehat{g}_i^b) \leftarrow \text{Enc}_{pp}(x_i^b, i)$$

for every $i \in H$ and $b \in \{0, 1\}$.

For a RARE in the common random string model, the public parameters pp are simply a random string and we require $\mathcal{H}$ -robustness for $\mathcal{H} = 2^{[m]}$ (arbitrary subsets).

For an honest-majority RARE, there are no public parameters, meaning that pp is empty, and we require $\mathcal{H}_{\text{maj}}$ -robustness, where

$$\mathcal{H}_{\text{maj}} := \{H \subseteq [m] : |H| > m/2\}.$$

Remark 1 (The group G). Throughout, the group G should be interpreted as a group ensemble $\{G_\lambda\}_{\lambda \in \mathbb{N}}$ with efficient representation and group operations. For simplicity of notation, we suppress λ and write simply G .

Two-Party Additive Randomized Encodings (2ARE). A 2ARE has the same syntax and correctness as a RARE with two honest parties; namely, $H = \{1, 2\}$, and security is required only against the evaluator. At the same time, we assume there are no public parameters and require not only indistinguishability, but also efficient simulation from the output. Namely, there exists a PPT simulator Sim such that, for every (x_1, x_2) , letting

$$\widehat{g}_i \leftarrow \text{Enc}(x_i, i) \quad \text{for } i \in \{1, 2\},$$

it holds that

$$\widehat{g}_1 + \widehat{g}_2 \approx_c \text{Sim}(f(x_1, x_2)).^4$$

Bitansky, Erabelli, Garg, and Ishai [BEGI26] construct such 2AREs unconditionally for NC^1 , and for all polynomial-size circuits assuming one-way functions. Their schemes have negligible correctness error. This error can be eliminated in the common random string model assuming one-way functions, or without a common random string assuming public-key encryption [BEG25].

Neighboring-Input Robustness. We use a relaxation of indistinguishability robustness in which the two honest-input vectors differ at only one coordinate. A similar relaxation was considered by Halevi et al. [HIJ+17] for non-interactive secure computation in the PKI model; we adapt it here to RAREs.

Definition 2 (Neighboring-input $\mathcal{H}$ -robustness). *Let $\mathcal{H} \subseteq 2^{[m]}$ be a family of subsets of parties. For $H \in \mathcal{H}$, two input vectors $\mathbf{x}_H^0 = (x_i^0)_{i \in H}$ and $\mathbf{x}_H^1 = (x_i^1)_{i \in H}$ are neighboring if there exists an index $j \in H$ such that*

$$x_i^0 = x_i^1 \quad \text{for every } i \in H \setminus \{j\}.$$

A RARE for f satisfies neighboring-input $\mathcal{H}$ -robustness if, for every $H \in \mathcal{H}$ and every pair of neighboring vectors $\mathbf{x}_H^0, \mathbf{x}_H^1$ such that

$$f_{H, \mathbf{x}_H^0} \equiv f_{H, \mathbf{x}_H^1},$$

the corresponding RARE distributions are computationally indistinguishable.

Lemma 1. *If every function in P/poly (respectively NC^1) admits a neighboring-input $\mathcal{H}$ -robust additive randomized encoding, then every function in P/poly (respectively NC^1) admits an $\mathcal{H}$ -robust additive randomized encoding (in the standard sense).*

Halevi et al. [HIJ+17] state an analogous lemma for non-interactive secure computation. Their proof seems to have a gap; In Appendix A we give an augmented proof adapted to RAREs.

2.2 Multi-String Tagged NIZK

We define multi-string tagged NIZK (MTNIZK), which can be seen as a multi-string NIZK [GO07] with a simulation-extractability property [Sah99, Gro06] for a designated party. Such a NIZK is associated with a number of parties m and a number of tags T . The (aggregate) reference string is a vector of independently generated random strings, one from each party $i \in [m]$. For any honest majority $H \subseteq [m]$, the honest strings can be programmed with a designated tag $j \in [T]$ so that proofs carrying tag j can be simulated. With overwhelming probability over the programmed honest strings, a witness can be extracted from every accepting proof carrying any other tag, regardless of the strings supplied by the remaining parties.

Let R be an NP relation, let λ be the security parameter, and write $[m] = \{1, \dots, m\}$.

Definition 3 (Multi-string tagged NIZK). *An (m, T) -multi-string-tagged NIZK for R consists of PPT algorithms*

$$\text{MTNIZK} := (\text{P}, \text{V}, \widetilde{\text{Gen}}, \widetilde{\text{P}}, \text{E}).$$

⁴For simplicity, we do not separate the 2ARE into a non-interactive part and a group part. Other than concrete efficiency considerations, this is without loss of generality, as the non-interactive part can be embedded in the group.

We abbreviate such a system as an (m, T) -MTNIZK. An ordinary setup samples independently at random

$$\rho_i \leftarrow \{0, 1\}^\lambda \quad \text{for each } i \in [m], \quad \boldsymbol{\rho} := (\rho_1, \dots, \rho_m).$$

Given a tag $t \in [T]$, a statement x , and a witness w satisfying $R(x, w) = 1$, the prover computes

$$\pi \leftarrow P(\boldsymbol{\rho}, t, x, w),$$

and the verifier outputs $V(\boldsymbol{\rho}, t, x, \pi) \in \{0, 1\}$.

The ordinary algorithms satisfy:

1. **Completeness.** For every $t \in [T]$ and every $(x, w) \in R$, independently sample $\rho_i \leftarrow \{0, 1\}^\lambda$ for each $i \in [m]$ and set $\boldsymbol{\rho} := (\rho_1, \dots, \rho_m)$. Then

$$\Pr [V(\boldsymbol{\rho}, t, x, P(\boldsymbol{\rho}, t, x, w)) = 1] = 1,$$

where the probability is over the sampled reference strings and the randomness of P .

For every tag $j \in [T]$ and every honest majority $H \subseteq [m]$, meaning $|H| > m/2$, the programming algorithm outputs

$$(\tilde{\boldsymbol{\rho}}_H, \tau) \leftarrow \widetilde{\text{Gen}}(1^\lambda, j, H),$$

where τ is a trapdoor for both simulation and extraction.

We require the following additional properties:

2. **Setup indistinguishability.** The programmed strings $\tilde{\boldsymbol{\rho}}_H$ are computationally indistinguishable from independently generated strings $\boldsymbol{\rho}_H$.
3. **Zero knowledge for tag j .** For every polynomial-length $(x, w) \in R$, every pair $(\tilde{\boldsymbol{\rho}}_H, \tau)$ in the support of $\widetilde{\text{Gen}}(1^\lambda, j, H)$, and every choice of the remaining strings $\tilde{\boldsymbol{\rho}}_{\bar{H}} = \{\tilde{\rho}_i\}_{i \notin H}$, letting $\tilde{\boldsymbol{\rho}} := (\tilde{\rho}_1, \dots, \tilde{\rho}_m)$, the distributions

$$\tilde{P}(\tilde{\boldsymbol{\rho}}, \tau, j, x) \quad \text{and} \quad P(\tilde{\boldsymbol{\rho}}, j, x, w)$$

are computationally indistinguishable. The scheme is ℓ -zero knowledge if the distinguishing advantage is at most $2^{-\ell(\lambda)} \cdot \text{negl}(\lambda)$.

4. **Extraction for all other tags.** With overwhelming probability over

$$(\tilde{\boldsymbol{\rho}}_H, \tau) \leftarrow \widetilde{\text{Gen}}(1^\lambda, j, H),$$

the following holds simultaneously for every choice of $\tilde{\boldsymbol{\rho}}_{\bar{H}}$, letting $\tilde{\boldsymbol{\rho}} := (\tilde{\boldsymbol{\rho}}_H, \tilde{\boldsymbol{\rho}}_{\bar{H}})$, every $t \in [T] \setminus \{j\}$, statement x , and proof π : if

$$V(\tilde{\boldsymbol{\rho}}, t, x, \pi) = 1,$$

then

$$w \leftarrow E(\tilde{\boldsymbol{\rho}}, \tau, t, x, \pi)$$

outputs a witness satisfying $R(x, w) = 1$.

Remark 2. Note that the size of each random string ρ is λ , independently of the size of the proven statement, which could be of arbitrary polynomial size in λ . We will sometimes be interested in ℓ -zero-knowledge for $\ell \gg |\rho|$ (this is why we do not simply define “subexponential zero knowledge”).

2.2.1 Instantiation

We give an instantiation from the following primitives:

1. a non-interactive pseudorandom commitment scheme Com for messages in $[T]$, with (perfect) binding for honestly generated commitments;
2. a non-interactive witness-indistinguishable proof system $\text{NIWI} := (\text{P}', \text{V}')$ for NP with subexponential witness indistinguishability; and
3. a witness encryption $\text{WE} := (\text{Enc}, \text{Dec})$ for NP with subexponential indistinguishability.

For a vector of (alleged) commitments $\mathbf{C} = (C_1, \dots, C_m)$ and a tag $j \in [T]$, let

$$\text{maj}(\mathbf{C}, = j)$$

denote the NP statement that more than half of the strings C_i are valid commitments to j . Namely, a witness consists of a set $S \subseteq [m]$ with $|S| > m/2$ and openings $(r_i)_{i \in S}$ such that $C_i = \text{Com}(j; r_i)$ for every $i \in S$. Similarly, let

$$\text{maj}(\mathbf{C}, \neq j)$$

denote the NP statement that more than half of the C_i are valid commitments to tags different from j . A witness now additionally contains tags $t_i \in [T] \setminus \{j\}$ such that $C_i = \text{Com}(t_i; r_i)$ for every $i \in S$.

Each reference string is an (alleged) commitment,

$$\rho_i = C_i.$$

The ordinary setup samples

$$C_i \leftarrow \{0, 1\}^\lambda.$$

To prove a statement x under tag j using a witness w with $R(x, w) = 1$, the prover computes a witness-encryption of w under the statement $\text{maj}(\mathbf{C}, \neq j)$:

$$ct \leftarrow \text{WE.Enc}(\text{maj}(\mathbf{C}, \neq j), w).$$

It then uses P' to produce a NIWI proof π' of the NP statement

$$\text{maj}(\mathbf{C}, = j) \quad \vee \quad (\exists w', s: R(x, w') = 1 \wedge ct = \text{WE.Enc}(\text{maj}(\mathbf{C}, \neq j), w'; s)).$$

The honest prover uses (w, s) as a witness for the second disjunct. The complete proof is

$$\pi := (ct, \pi').$$

The verifier accepts if V' accepts π' .

To program the strings of an honest majority H at tag j , for every $i \in H$ sample an opening r_i and set

$$\tilde{C}_i := \text{Com}(j; r_i),^5 \quad \tilde{\rho}_i := \tilde{C}_i.$$

The trapdoor is $\tau := (j, H, (r_i)_{i \in H})$.

Setup indistinguishability follows from the pseudorandomness of the commitments.

⁵We assume without loss of generality that $|j| \leq O(\log \lambda)$, so the entire commitment has length λ .

The proof simulator $\tilde{P}(\tilde{\rho}, \tau, j, x)$ encrypts an all-zero string of the prescribed witness length under the instance $\text{maj}(\mathbf{C}, \neq j)$. It then uses the openings in τ , as the witness for $\text{maj}(\mathbf{C}, = j)$ for computing the NIWI π' and outputs the resulting proof (ct, π') . Zero knowledge follows from witness indistinguishability of the NIWI and semantic security of the witness encryption; indeed, $\text{maj}(\mathbf{C}, \neq j)$ is false because a majority of the binding commitments open to j . Furthermore, by choosing the security parameters of the NIWI and witness encryption to be sufficiently large polynomials in $|\ell(\lambda)|$, their subexponential security implies $2^{-\ell}$ -zero knowledge.

For extraction, consider an accepting proof (ct, π') carrying a tag $t \neq j$. The openings in τ give a witness ω_t for $\text{maj}(\mathbf{C}, \neq t)$, and the extractor $E(\tilde{\rho}, \tau, t, x, (ct, \pi'))$ outputs

$$w := \text{WE.Dec}(ct, \omega_t).$$

Except with negligible probability over the programmed commitments, binding rules out $\text{maj}(\mathbf{C}, = t)$. NIWI soundness therefore implies that the extracted w is a valid witness for x . Since H is a majority, this conclusion holds for every choice of the remaining reference strings and simultaneously for all tags, statements, and accepting proofs, as required.

Instantiations of the Required Primitives. The required non-interactive pseudorandom commitments can be obtained from Naor's commitment scheme [Nao91]; honest commitments are binding and pseudorandom. NIWI proof systems with deterministic verification can be constructed from bilinear-map assumptions [GOS06], or from ZAPs together with an appropriate derandomization assumption [BOV03]. NIWIs also follow from indistinguishability obfuscation and honestly-binding dense commitments [BP15], which are known either from DDH or LWE. The resulting proof system, however, has an MA-style randomized verifier. For the sake of simplicity, we restrict attention to deterministic verification. Finally, witness encryption for NP follows from indistinguishability obfuscation [GGSW13]. We assume subexponentially secure instantiations of the NIWI and witness encryption, as quantified above.

2.2.2 Single-String Tagged NIZK

The setup-free honest-majority construction will use the multi-string system defined above. The CRS-model construction of Section 3, which we present first, uses the following single-string special (simplified) case.

Let R be an NP relation and let T be the number of tags.

Definition 4 (Single-string tagged NIZK). *A single-string tagged NIZK for R with tag space $[T]$ consists of PPT algorithms*

$$\text{TNIZK} := (\mathbf{P}, \mathbf{V}, \widetilde{\text{Gen}}, \tilde{P}, \mathbf{E}).$$

The ordinary reference string is sampled uniformly as

$$\rho \leftarrow \{0, 1\}^\lambda.$$

Given a tag $t \in [T]$, a statement x , and a witness w satisfying $R(x, w) = 1$, the prover computes

$$\pi \leftarrow \mathbf{P}(\rho, t, x, w),$$

and the verifier outputs $\mathbf{V}(\rho, t, x, \pi) \in \{0, 1\}$.

The ordinary algorithms satisfy perfect completeness: for every $t \in [T]$ and $(x, w) \in R$,

$$\Pr [\mathbf{V}(\rho, t, x, \mathbf{P}(\rho, t, x, w)) = 1] = 1,$$

where the probability is over the uniform reference string and the prover's randomness.

For every designated tag $j \in [T]$, the programmed setup algorithm outputs

$$(\tilde{\rho}, \tau) \leftarrow \widetilde{\text{Gen}}(1^\lambda, j),$$

where τ is used for both simulation and extraction. The following properties are required.

1. **Setup indistinguishability.** The distributions $\tilde{\rho}$ and ρ are computationally indistinguishable.
2. **Zero knowledge for the programmed tag.** For every polynomial-length $(x, w) \in R$ and every $(\tilde{\rho}, \tau)$ in the support of $\widetilde{\text{Gen}}(1^\lambda, j)$, the distributions

$$P(\tilde{\rho}, j, x, w) \quad \text{and} \quad \tilde{P}(\tilde{\rho}, \tau, j, x)$$

are computationally indistinguishable.

3. **Extraction for every other tag.** With overwhelming probability over $(\tilde{\rho}, \tau) \leftarrow \widetilde{\text{Gen}}(1^\lambda, j)$, the following holds simultaneously for every $t \in [T] \setminus \{j\}$, statement x , and proof π : if

$$V(\tilde{\rho}, t, x, \pi) = 1,$$

then

$$w \leftarrow E(\tilde{\rho}, \tau, t, x, \pi)$$

satisfies $R(x, w) = 1$.

Instantiation. For our CRS-model RARE construction (Section 3), standard computational zero knowledge is sufficient; we do not require the stronger ℓ -zero-knowledge guarantee. Consequently, the witness encryption and the witness-indistinguishable proof used in the preceding instantiation need only have ordinary polynomial security. Moreover, the NIWI can be replaced by a ZAP [DN00]. Concretely, the single reference string is parsed as

$$\rho = (C, Z),$$

where C is an alleged commitment and Z is the first message of a ZAP. The prover forms the witness-encryption ciphertext as above and uses Z to prove the same disjunction. To program the string at tag j , the commitment C is generated as a commitment to j , while Z is generated honestly. Simulation and extraction, as well as their analyses, are otherwise identical to those of the multi-string construction.

This ZAP-based instantiation requires only polynomially secure witness encryption and ZAP witness indistinguishability. In particular, ZAPs can be constructed from indistinguishability obfuscation and one-way functions [BP15].

2.3 Multi-Key Homomorphic Encryption

We define secret-key multi-key homomorphic encryption in the common random string model, with computationally simulatable partial decryption.

Let $\mathcal{C} = \{\mathcal{C}_\lambda\}_{\lambda \in \mathbb{N}}$ be a circuit class, where every $C \in \mathcal{C}_\lambda$ is a polynomial-size circuit

$$C: (\{0, 1\}^n)^m \rightarrow \{0, 1\}^*.$$

Definition 5 (Secret-key multi-key homomorphic encryption). *A secret-key multi-key homomorphic encryption scheme for $\mathcal{C}$, with plaintext length $n = n(\lambda)$ and $m = m(\lambda)$ parties, consists of algorithms*

$$\text{MHE} := (\text{Enc}, \text{Eval}, \text{PDec}, \text{Dec}, \widetilde{\text{PDec}}).$$

The algorithms Enc , Eval , and Dec are DPT, whereas PDec and $\widetilde{\text{PDec}}$ are PPT.

The common random string is sampled uniformly as

$$\eta \leftarrow \{0, 1\}^\lambda.$$

Each user independently samples a uniformly random secret key

$$\text{sek}_i \leftarrow \{0, 1\}^\lambda$$

and encrypts a plaintext $x_i \in \{0, 1\}^n$ as

$$\text{ct}_i := \text{Enc}_\eta(\text{sek}_i, x_i).$$

Given any $C \in \mathcal{C}_\lambda$ and a fresh ciphertext vector $\mathbf{ct} := (ct_1, \dots, ct_m)$, possibly encrypted under different independently generated keys, anyone can compute

$$\mathbf{ct}^* := \text{Eval}_\eta(C, \mathbf{ct}).$$

Each user partially decrypts the evaluated ciphertext using its corresponding secret key:

$$d_i \leftarrow \text{PDec}_\eta(\text{sek}_i, \mathbf{ct}^*) \quad \text{for } i \in [m].$$

Given the vector of partial decrypts $\mathbf{d} := (d_1, \dots, d_m)$, the decryption algorithm outputs

$$y := \text{Dec}_\eta(\mathbf{d}).$$

For every $i \in [m]$, a simulated partial decryption for user i can be generated from the other users' secret keys, the evaluated ciphertext, and the output:

$$\tilde{d}_i \leftarrow \widetilde{\text{PDec}}_\eta(i, \mathbf{sek}_{-i}, \mathbf{ct}^*, y).$$

The scheme satisfies the following properties.

1. **Perfect correctness.** *For every key vector and plaintext vector*

$$\mathbf{sek} := (\text{sek}_1, \dots, \text{sek}_m) \in (\{0, 1\}^\lambda)^m, \quad \mathbf{x} := (x_1, \dots, x_m) \in (\{0, 1\}^n)^m,$$

and every $C \in \mathcal{C}_\lambda$, let

$$\eta \leftarrow \{0, 1\}^\lambda, \quad \text{ct}_i := \text{Enc}_\eta(\text{sek}_i, x_i) \quad \text{for } i \in [m], \quad \mathbf{ct}^* := \text{Eval}_\eta(C, \mathbf{ct}),$$

where $\mathbf{ct} := (ct_1, \dots, ct_m)$, and let

$$d_i \leftarrow \text{PDec}_\eta(\text{sek}_i, \mathbf{ct}^*) \quad \text{for } i \in [m], \quad \mathbf{d} := (d_1, \dots, d_m).$$

Then

$$\Pr[\text{Dec}_\eta(\mathbf{d}) = C(\mathbf{x})] = 1,$$

where the probability is over the sampling of η and the randomness of the partial-decryption algorithms.

2. **Semantic security.** For every pair of plaintexts $x_0, x_1 \in \{0, 1\}^n$, the distributions

$$(\eta, \text{Enc}_\eta(\text{sek}, x_0)) \quad \text{and} \quad (\eta, \text{Enc}_\eta(\text{sek}, x_1)),$$

are computationally indistinguishable, where $\eta \leftarrow \{0, 1\}^\lambda$ and $\text{sek} \leftarrow \{0, 1\}^\lambda$.

3. **Simulation of partial decryption.** For every $\eta \in \{0, 1\}^\lambda$, every $i \in [m]$, and every key and plaintext vector

$$\text{sek} := (\text{sek}_1, \dots, \text{sek}_m) \in (\{0, 1\}^\lambda)^m, \quad \mathbf{x} := (x_1, \dots, x_m) \in (\{0, 1\}^n)^m,$$

let $ct_j := \text{Enc}_\eta(\text{sek}_j, x_j)$ for $j \in [m]$ and $\mathbf{ct} := (ct_1, \dots, ct_m)$. For every $C \in \mathcal{C}_\lambda$ and evaluated ciphertext

$$ct^* := \text{Eval}_\eta(C, \mathbf{ct}),$$

setting $y := C(\mathbf{x})$, the distributions

$$\text{PDec}_\eta(\text{sek}_i, ct^*) \quad \text{and} \quad \widetilde{\text{PDec}}_\eta(i, \text{sek}_{-i}, ct^*, y)$$

are computationally indistinguishable.

We say that the scheme is subexponentially secure if there exists a constant $\varepsilon > 0$ such that every polynomial-size circuit distinguisher has advantage at most $2^{-\lambda^\varepsilon}$ between each pair of distributions in the semantic security and partial-decryption simulation properties above. We say that it has subexponential decryption simulation accuracy if this only holds for partial decryption alone (see relevant discussion below in the instantiation paragraph).

Instantiation. An MHE scheme for general functions with statistical subexponential simulation accuracy is constructed in [MW16] from LWE with a common random string. Assuming subexponential hardness of LWE, it also has subexponential semantic security. We note that the latter work, in fact, constructs *public key* MHE. To simplify notation, we restrict attention to secret key MHE, which directly follows from public-key MHE (by including the public-key in the ciphertext).

Dodis et al. [DHRW16] construct a *spooky encryption* scheme for NC^1 under DDH, that is completely setup free. This yields an MHE that is setup free. The resulting MHE also has a strong deterministic partial-decryption simulation property: both partial-decryption algorithms are deterministic and, for consistent inputs, satisfy

$$\text{PDec}_\eta(\text{sek}_i, ct^*) = \widetilde{\text{PDec}}_\eta(i, \text{sek}_{-i}, ct^*, y).$$

Other setup-free MHE constructions (cf. [AJJM20]) satisfy a weaker simulation property that does not seem to suffice for our purpose.

2.4 Puncturable Pseudorandom Functions

We recall the standard notion of a puncturable pseudorandom function (puncturable PRF) [BGI14, BW13, KPTZ13].

Definition 6 (Puncturable pseudorandom function). Let $d = d(\lambda)$ and $\ell = \ell(\lambda)$ be polynomially bounded. A puncturable PRF with domain $\{0, 1\}^d$ and range $\{0, 1\}^\ell$ consists of deterministic polynomial-time algorithms

$$\text{PPRF} := (\text{Eval}, \text{Punc}).$$

A key is sampled uniformly as

$$K \leftarrow \{0,1\}^\lambda,$$

and evaluation is written as

$$F_K(x) := \text{Eval}(K, x).$$

Given a point $x^* \in \{0,1\}^d$, the puncturing algorithm outputs a key

$$K\{x^*\} \leftarrow \text{Punc}(K, x^*)$$

that permits evaluation at every point other than x^* . The scheme satisfies the following properties.

1. **Functionality preservation.** For every $\lambda \in \mathbb{N}$, every key $K \in \{0,1\}^\lambda$, and every $x^*, x \in \{0,1\}^d$ with $x \neq x^*$, letting $K\{x^*\} := \text{Punc}(K, x^*)$, we have

$$\text{Eval}(K\{x^*\}, x) = F_K(x).$$

2. **Pseudorandomness at the punctured point.** For every sequence $x^* = x^*(\lambda) \in \{0,1\}^d$, the distributions

$$(K\{x^*\}, F_K(x^*)) \quad \text{and} \quad (K\{x^*\}, U_\ell)$$

are computationally indistinguishable, where $K \leftarrow \{0,1\}^\lambda$, $K\{x^*\} \leftarrow \text{Punc}(K, x^*)$, and $U_\ell \leftarrow \{0,1\}^\ell$ is uniform and independent.

We say that the scheme is subexponentially secure if there exists a constant $\varepsilon > 0$ such that every polynomial-size circuit distinguisher has advantage at most $2^{-\lambda^\varepsilon}$ between these distributions.

PPRFs are known based on one-way functions; see, for example, [BGI14, BW13, KPTZ13].

2.5 Indistinguishability Obfuscation

We recall the standard notion of indistinguishability obfuscation [BGI+01].

Definition 7 (Indistinguishability obfuscation). An indistinguishability obfuscator for all circuits is a PPT algorithm iO that, given a Boolean circuit C , outputs a circuit

$$\tilde{C} \leftarrow \text{iO}(C).$$

It satisfies the following properties.

1. **Correctness.** For every Boolean circuit C and every input x in the domain of C , if $\tilde{C} \leftarrow \text{iO}(C)$, then

$$\Pr[\tilde{C}(x) = C(x)] = 1.$$

2. **Indistinguishability.** For every pair of circuit ensembles $C_0 = \{C_{0,\lambda}\}_{\lambda \in \mathbb{N}}$ and $C_1 = \{C_{1,\lambda}\}_{\lambda \in \mathbb{N}}$ such that, for every λ , the circuits $C_{0,\lambda}$ and $C_{1,\lambda}$ have the same size and compute the same function, the distributions

$$\text{iO}(C_0) \quad \text{and} \quad \text{iO}(C_1)$$

are computationally indistinguishable.

We say that the scheme is subexponentially secure if there exists a constant $\varepsilon > 0$ such that every polynomial-size circuit distinguisher has advantage at most $2^{-\lambda^\varepsilon}$ between these distributions.

Jain, Lin, and Sahai construct (subexponentially secure) iO assuming subexponential security of LPN over prime fields, PRGs in NC^0 with polynomial stretch, and DLIN over symmetric bilinear groups [JLS22].

2.5.1 Nondeterministic Probabilistic iO

We use the following Nondeterministic pIO Lemma, which follows from [CLTV15].

Lemma 2 (Nondeterministic pIO). *Let $R = \{R_\lambda\}_{\lambda \in \mathbb{N}}$ be a polynomial-size computable NP relation where $R \subseteq X \times W$ for $X = \{X_\lambda\}_{\lambda \in \mathbb{N}}$, $W = \{W_\lambda\}_{\lambda \in \mathbb{N}}$ and let $L = \{L_\lambda\}_{\lambda \in \mathbb{N}}$ be the corresponding language. Let $\mathcal{P} = \{\mathcal{P}_\lambda\}_{\lambda \in \mathbb{N}}$ be an ensemble of polynomial-size circuit samplers such that $\mathcal{P}_\lambda$ outputs a pair (P_0, P_1) of equal-size probabilistic circuits defined over X_λ .*

For a probabilistic circuit P , define the verification-augmented probabilistic circuit

$$P^R(x, w; r) := \begin{cases} P(x; r), & \text{if } R_\lambda(x, w) = 1, \\ \perp, & \text{otherwise.} \end{cases}$$

For a PPRF key K , let $P^R[K]$ be the derandomized variant, defined by

$$P^R[K](x, w) := \begin{cases} P(x; \text{PPRF.Eval}(K, x)), & \text{if } R_\lambda(x, w) = 1, \\ \perp, & \text{otherwise.} \end{cases}$$

Assume that for every polynomial-size distinguisher D and every $x \in L_\lambda$, it holds that

$$\left| \Pr_{\substack{(P_0, P_1) \leftarrow \mathcal{P}_\lambda \\ r \leftarrow \{0,1\}^{\ell(\lambda)}}} [D(P_0, P_1, P_0(x; r)) = 1] - \Pr_{\substack{(P_0, P_1) \leftarrow \mathcal{P}_\lambda \\ r \leftarrow \{0,1\}^{\ell(\lambda)}}} [D(P_0, P_1, P_1(x; r)) = 1] \right| \leq \frac{\text{negl}(\lambda)}{|L_\lambda|}.$$

Assume further that iO and a puncturable PRF PPRF with domain X_λ are each secure with advantage at most $\text{negl}(\lambda)/|L_\lambda|$. Then, for $(P_0, P_1) \leftarrow \mathcal{P}_\lambda$ and $K \leftarrow \{0,1\}^\lambda$, the distributions

$$(P_0, P_1, \text{iO}(P_0^R[K])) \quad \text{and} \quad (P_0, P_1, \text{iO}(P_1^R[K]))$$

are computationally indistinguishable.

2.6 Programmable Signatures

A programmable signature scheme admits an alternative key-generation algorithm that programs a verification key for one designated message. The programmed key and its designated signature look like an honestly generated verification key and signature, but the programmed key rejects every signature on any other message. The notion can be viewed as a simplified version of splittable signatures [KLW15].

Definition 8 (Programmable signature). *A programmable signature scheme for a message space $\mathcal{M}$ consists of PPT algorithms Gen and $\widetilde{\text{Gen}}$, and DPT algorithms Sig and Ver , written as*

$$\text{PSIG} := (\text{Gen}, \text{Sig}, \text{Ver}, \widetilde{\text{Gen}}).$$

The ordinary algorithms have the usual signature syntax:

$$(ssk, vk) \leftarrow \text{Gen}(1^\lambda), \quad \sigma := \text{Sig}(ssk, \mu), \quad \text{Ver}(vk, \mu, \sigma) \in \{0, 1\}.$$

Given a designated message $\mu \in \mathcal{M}$, the programmed key-generation algorithm outputs a programmed verification key and a signature on μ :

$$(vk\{\mu\}, \sigma\{\mu\}) \leftarrow \widetilde{\text{Gen}}(1^\lambda, \mu).$$

The scheme satisfies the following properties.

1. **Correctness.** For every $\mu \in \mathcal{M}$, if $(ssk, vk) \leftarrow \text{Gen}(1^\lambda)$ and $\sigma := \text{Sig}(ssk, \mu)$, then

$$\Pr[\text{Ver}(vk, \mu, \sigma) = 1] = 1.$$

2. **Programming indistinguishability.** For every $\mu \in \mathcal{M}$, the distributions

$$(vk\{\mu\}, \sigma\{\mu\}) \quad \text{and} \quad (vk, \sigma)$$

are computationally indistinguishable, where

$$(vk\{\mu\}, \sigma\{\mu\}) \leftarrow \widetilde{\text{Gen}}(1^\lambda, \mu), \quad (ssk, vk) \leftarrow \text{Gen}(1^\lambda), \quad \sigma := \text{Sig}(ssk, \mu).$$

3. **Unique verification.** For every $\mu \in \mathcal{M}$, with overwhelming probability over

$$(vk\{\mu\}, \sigma\{\mu\}) \leftarrow \widetilde{\text{Gen}}(1^\lambda, \mu),$$

for every $\mu' \neq \mu$ and every purported signature σ' , we have

$$\text{Ver}(vk\{\mu\}, \mu', \sigma') = 0.$$

2.6.1 Instantiation

Programmable signatures can be constructed from a statistically binding, computationally hiding non-interactive commitment scheme Com and a NIZK proof system $\text{NIZK} := (\text{Gen}, \text{P}, \text{V})$ for NP. Let $\perp \notin \mathcal{M}$ be a distinguished symbol. A real verification key (ρ, C) consists of a NIZK reference string $\rho \leftarrow \text{NIZK.Gen}(1^\lambda)$ and a commitment $C := \text{Com}(\perp; r_C)$ to $\perp$. The signing key is $ssk := (r_C, r_\pi)$, where r_C is the commitment randomness and r_π consists of uniformly sampled coins for the NIZK prover. A signature on a message μ is a NIZK proof of the statement

$$\exists r'_C: C = \text{Com}(\perp; r'_C) \quad \vee \quad C = \text{Com}(\mu; r'_C).$$

That is, the verification key is $vk := (\rho, C)$ and the signature is

$$\pi := \text{NIZK.P}(\rho, (C, \mu), r_C; r_\pi),$$

where r_C is the witness and the prover coins r_π are fixed by the signing key. Thus, signing is deterministic given ssk and μ . The verification algorithm simply verifies the proof π and is also deterministic.

A programmed verification key and signature for μ are generated by sampling an ordinary NIZK reference string and NIZK prover coins r_π , setting $C := \text{Com}(\mu; r_C)$, and producing the corresponding NIZK proof using r_C as a witness for the second disjunct and r_π as the prover coins.

Correctness follows because an honest signer knows an opening of C to $\perp$. To prove programming indistinguishability, first replace the honest NIZK proof in the real distribution by a simulated proof. Once the proof is simulated, replace the commitment to $\perp$ by a commitment to μ using computational hiding, and then replace the simulated proof by an honest proof using the opening to μ .

Finally, consider a key programmed for μ . Except with negligible probability over the commitment and NIZK reference string, statistical binding and NIZK soundness imply that no proof can be accepted for a message $\mu' \neq \mu$: such a proof would assert that the same commitment opens either to $\perp$ or to μ' , contradicting its binding to μ . This gives unique verification.

2.7 Non-Interactive Commitments

We recall the standard notion of a non-interactive commitment scheme.

Definition 9 (Non-interactive commitment). *Let $n = n(\lambda)$ be polynomially bounded. A non-interactive commitment scheme is a PPT algorithm Com that, given a message $x \in \{0, 1\}^n$, outputs a commitment C . The scheme satisfies the following properties.*

1. **Computational hiding.** *For every pair of length n message (ensembles) x_0, x_1*

$$\text{Com}(x_0) \quad \text{and} \quad \text{Com}(x_1)$$

are computationally indistinguishable.

2. **Perfect binding.** *There do not exist $x_0 \neq x_1$ and random coins r_0, r_1 such that*

$$\text{Com}(x_0; r_0) = \text{Com}(x_1; r_1).$$

Non-interactive commitments are known from a variety of assumptions. In particular, they follow from iO and one-way functions [BPW16], which are assumed in this.

3 RARE in the Common Random String Model

We construct RAREs in the common random string (CRS) model secure against every corruption pattern.

Theorem 1 (restated). *Assuming iO and LWE, both with subexponential security, there exist RAREs in the common random string model for all efficiently computable functions, secure against arbitrary corruption patterns.*

Remark 3 (Generic Primitives and Subexponential Security). The list of required generic primitives can be found below in Section 3.1. Each of these primitives can be constructed from iO and one-way functions, except for MHE, for which we assume LWE.

We also note that if the underlying MHE satisfy a stronger deterministic partial decryption property (and plain polynomial semantic security), then the assumptions can be weakened to polynomial (rather than subexponential) iO and PPRF. However, at the moment, MHE satisfying this property is only known based on subexponential iO and DDH, and only for NC¹.

3.1 Construction

We now describe all components of the construction. Its security analysis will be given separately.

Ingredients. The construction uses the following primitives:

1. a computationally hiding, perfectly binding non-interactive commitment scheme Com for messages in $\{0, 1\}^n$, using randomness in $\{0, 1\}^\lambda$;
2. a pseudorandom generator

$$G_{\text{sek}}: \{0, 1\}^\lambda \longrightarrow \{0, 1\}^{\tilde{\lambda}},$$

whose output is a parameter $\tilde{\lambda}$ to be specified below;

3. a single-string tagged NIZK TNIZK with tag space $[m]$, for the relation

$$R_{\text{ct}} := \{((\eta, C, ct); (x, r, s)) : \eta \in \{0, 1\}^{\tilde{\lambda}}, \quad x \in \{0, 1\}^n, \quad r, s \in \{0, 1\}^\lambda, \\ C = \text{Com}(x; r), \quad ct = \text{MHE.Enc}_\eta(G_{\text{sek}}(s), x)\},$$

4. a programmable signature scheme PSIG ;
5. a 2ARE for the authentication function Auth defined below;
6. a secret-key multi-key homomorphic encryption scheme MHE in the common random string model for a circuit class $\mathcal{C} \ni f$ with subexponential decryption simulation accuracy, instantiated with security parameter $\tilde{\lambda}$;
7. a subexponentially secure indistinguishability obfuscator iO ; and
8. a subexponentially secure puncturable PRF PPRF .

For simplicity, suppose that MHE , iO , and PPRF have a common subexponential security exponent $\varepsilon > 0$; if their exponents differ, take ε to be the smallest of the three. We choose $\tilde{\lambda}$ so that

$$\tilde{\lambda}^\varepsilon \geq m(n + 2\lambda).$$

Setup. The RARE setup algorithm samples uniformly random strings

$$\rho \leftarrow \{0, 1\}^\lambda, \quad \eta \leftarrow \{0, 1\}^{\tilde{\lambda}},$$

and publishes

$$pp := (\rho, \eta).$$

Both the encoder and decoder receive pp .

2ARE Authentication of Commitments. We use 2ARE to allow parties to authenticate one another's input commitments. For $i \neq j$, the key $vk_{i,j}$ is used by party i to authenticate party j 's commitment C_j , and the corresponding signature is $\sigma_{i,j}$. The mechanism is specified in Figure 1.

Figure 1: Authentication of commitments using 2ARE

For each ordered pair (i, j) with $i \neq j$, party i independently generates the key pair

$$(ssk_{i,j}, vk_{i,j}) \leftarrow \text{PSIG.Gen}(1^\lambda).$$

Figure 1 (continued)

For every unordered pair $\{i, j\}$, define

$$\begin{aligned} & \text{Auth}_{i,j}((ssk_{i,j}, C_i), (ssk_{j,i}, C_j)) \\ & := (\text{PSIG.Sig}(ssk_{i,j}, C_j), \text{PSIG.Sig}(ssk_{j,i}, C_i)) \\ & =: (\sigma_{i,j}, \sigma_{j,i}). \end{aligned}$$

The parties run the two local encoders of a 2ARE for $\text{Auth}_{i,j}$. Party i uses local input

$$u_{i,j} := (ssk_{i,j}, C_i)$$

and obtains

$$\hat{g}_{i,j} \leftarrow \text{2ARE.Enc}_{i,j}^{(i)}(u_{i,j});$$

party j acts symmetrically to obtain $\hat{g}_{j,i}$. The pairwise decoder recovers the two signatures above from

$$\hat{g}_{i,j} + \hat{g}_{j,i}.$$

The group of the resulting RARE is the product of the 2ARE groups over the $\binom{m}{2}$ unordered pairs. Party i 's group component $\hat{g}_i$ places $\hat{g}_{i,j}$ in coordinate $\{i, j\}$ for every $j \neq i$ and the identity element in all other coordinates. Consequently,

$$\left(\sum_{i \in [m]} \hat{g}_i \right)_{\{i,j\}} = \hat{g}_{i,j} + \hat{g}_{j,i} \quad \text{for every } \{i, j\}.$$

Obfuscated Evaluation Program. For a fixed party i , write

$$\begin{aligned} \mathbf{vk}_{i,-i} &:= (vk_{i,j})_{j \neq i}, & \boldsymbol{\sigma}_{i,-i} &:= (\sigma_{i,j})_{j \neq i}, & \mathbf{C}_{-i} &:= (C_j)_{j \neq i}, \\ \mathbf{ct}_{-i} &:= (ct_j)_{j \neq i}, & \boldsymbol{\pi}_{-i} &:= (\pi_j)_{j \neq i}. \end{aligned}$$

Party i publishes an obfuscation of the program

$$\text{EV}_i[f, \rho, \eta, \mathbf{vk}_{i,-i}, ct_i, sek_i, K_i](\mathbf{C}_{-i}, \mathbf{ct}_{-i}, \boldsymbol{\sigma}_{i,-i}, \boldsymbol{\pi}_{-i}),$$

specified in Figure 2.

Figure 2: Homomorphic evaluation program EV_i

Hardwired:

- f , ρ , and η ;
- $\mathbf{vk}_{i,-i}$;
- party i 's ciphertext ct_i and secret key sek_i ; and
- a puncturable-PRF key K_i .

Input:

- $\mathbf{C}_{-i}$ and $\mathbf{ct}_{-i}$, the other parties' commitments and ciphertexts;

Figure 2 (continued)

- $\sigma_{i,-i}$, the signatures on the commitments; and
- π_{-i} , their tagged proofs.

Computation:

1. **Verify signatures and proofs:** For every $j \neq i$, check

$$\text{PSIG.Ver}(vk_{i,j}, C_j, \sigma_{i,j}) = 1$$

and

$$\text{TNIZK.V}(\rho, j, (\eta, C_j, ct_j), \pi_j) = 1.$$

If any check fails, output $\perp$.

2. **Evaluate:** Let $ct := (ct_1, \dots, ct_m)$ and compute

$$ct^* := \text{MHE.Eval}_\eta(f, ct).$$

3. **Partially decrypt:** Derive

$$r_i := \text{PPRF.Eval}(K_i, (C_{-i}, ct_{-i}))$$

and output

$$d_i \leftarrow \text{MHE.PDec}_\eta(sek_i, ct^*; r_i).$$

The complete RARE encoder and decoder are given in Figures 3 and 4, respectively.

Figure 3: RARE encoder

Given $pp = (\rho, \eta)$, party index i , and input x_i :

1. **Commit to the input:** Sample commitment randomness

$$r_i \leftarrow \{0, 1\}^\lambda$$

and set

$$C_i := \text{Com}(x_i; r_i).$$

2. **Generate the MHE key and ciphertext:** Sample

$$s_i \leftarrow \{0, 1\}^\lambda, \quad sek_i := G_{\text{sek}}(s_i),$$

and set

$$ct_i := \text{MHE.Enc}_\eta(sek_i, x_i).$$

3. **Prove consistency:** Generate the tagged proof

$$\pi_i \leftarrow \text{TNIZK.P}(\rho, i, (\eta, C_i, ct_i), (x_i, r_i, s_i)).$$

Figure 3 (continued)

4. **Generate keys:** For every $j \neq i$, generate

$$(ssk_{i,j}, vk_{i,j}) \leftarrow \text{PSIG.Gen}(1^\lambda),$$

and sample

$$K_i \leftarrow \{0, 1\}^{\tilde{\lambda}}.$$

5. **Obfuscate the evaluation program:** Compute the padded obfuscation

$$\widetilde{\text{EV}}_i \leftarrow \text{iO}(\text{EV}_i[f, \rho, \eta, vk_{i,-i}, ct_i, sek_i, K_i]).$$

6. **Encode the authentication functionality:** For every $j \neq i$, set

$$u_{i,j} := (ssk_{i,j}, C_i)$$

and compute the local 2ARE component

$$\widehat{g}_{i,j} \leftarrow \text{2ARE.Enc}_{i,j}^{(i)}(u_{i,j}),$$

and let $\widehat{g}_i$ be the corresponding product-group element.

7. **Output the local encoding:** Set

$$\widehat{z}_i := (C_i, ct_i, \pi_i, \widetilde{\text{EV}}_i),$$

and output $(\widehat{z}_i, \widehat{g}_i)$.

Figure 4: RARE decoder

The evaluator is given $pp = (\rho, \eta)$ and the non-interactive components, parsed as

$$\widehat{\mathbf{z}} = (C, ct, \pi, \widetilde{\text{EV}})$$

together with the aggregate product-group element $\widehat{g} := \sum_{i \in [m]} \widehat{g}_i$. It proceeds as follows:

1. **Decode the 2AREs:** For every unordered pair $\{i, j\}$, compute

$$(\sigma_{i,j}, \sigma_{j,i}) := \text{2ARE.Dec}_{i,j}(\widehat{g}_{\{i,j\}}).$$

2. **Generate partial decryptions:** For every $i \in [m]$, compute

$$d_i \leftarrow \widetilde{\text{EV}}_i(C_{-i}, ct_{-i}, \sigma_{i,-i}, \pi_{-i}).$$

3. **Decrypt:** Let $\mathbf{d} := (d_1, \dots, d_m)$ and output

$$y := \text{MHE.Dec}_\eta(\mathbf{d}).$$

Correctness. When all parties are honest, every pairwise 2ARE decodes to the required commitment signatures, and every TNIZK proof verifies that the ciphertext encrypts the committed input. Hence all evaluation programs evaluate f on the same honestly generated ciphertext vector and produce the corresponding partial decryptions. Perfect correctness of MHE implies that the decoder outputs $f(x_1, \dots, x_m)$ with probability one.

3.2 Security Analysis

Proposition 1. *The CRS-model construction satisfies neighboring-input $2^{[m]}$ -robustness.*

Proof. Fix $H \subseteq [m]$ and neighboring input vectors $\mathbf{x}_H^0 = (x_i^0)_{i \in H}$ and $\mathbf{x}_H^1 = (x_i^1)_{i \in H}$ such that

$$f_{H, \mathbf{x}_H^0} \equiv f_{H, \mathbf{x}_H^1}.$$

The case $H = \emptyset$ is immediate, so assume $H \neq \emptyset$. Let $j \in H$ be the unique party on whose input the two vectors may differ; thus, $x_i^0 = x_i^1$ for every $i \in H \setminus \{j\}$.

For an honest input vector $\mathbf{u}_H = (u_i)_{i \in H}$, define the distribution of the evaluator's view by

$$\text{View}_{\text{crs}}(H, \mathbf{u}_H) := (pp, \widehat{\mathbf{z}}_H, \widehat{\mathbf{g}}_H), \quad \widehat{\mathbf{g}}_H := \sum_{i \in H} \widehat{g}_i,$$

where $pp \leftarrow \text{Setup}(1^\lambda)$ and, independently for every $i \in H$,

$$(\widehat{z}_i, \widehat{g}_i) \leftarrow \text{Enc}_{pp}(u_i, i).$$

Equivalently, $\text{View}_{\text{crs}}(H, \mathbf{u}_H)$ consists of the public parameter, all non-interactive components produced by the parties in H , and the sum of their group components. Our goal is to prove

$$\text{View}_{\text{crs}}(H, \mathbf{x}_H^0) \approx_c \text{View}_{\text{crs}}(H, \mathbf{x}_H^1).$$

We establish this claim through a sequence of hybrid distributions. In the description of each hybrid, components changed relative to the preceding hybrid will be colored **red**.

For each $b \in \{0, 1\}$, define hybrids $\text{Hyb}_0^b, \dots, \text{Hyb}_6^b$. We will show that each consecutive pair is computationally indistinguishable and that the final distribution is independent of b .

Hybrid 0: Real. This is the real distribution

$$\text{Hyb}_0^b := \text{View}_{\text{crs}}(H, \mathbf{x}_H^b).$$

Hybrid 1: Simulating Non-Corrupted 2ARE. For every unordered pair $\{\alpha, \beta\} \subseteq H$, let

$$\boldsymbol{\sigma}_{\{\alpha, \beta\}} := (\sigma_{\alpha, \beta}, \sigma_{\beta, \alpha})$$

be the output of the corresponding authentication function. In coordinate $\{\alpha, \beta\}$ of the aggregate group element, replace

$$\widehat{g}_{\alpha, \beta} + \widehat{g}_{\beta, \alpha} \quad \text{by} \quad \widetilde{g}_{\{\alpha, \beta\}} \leftarrow \text{2ARE.Sim}(\boldsymbol{\sigma}_{\{\alpha, \beta\}}).$$

The distributions Hyb_0^b and Hyb_1^b are computationally indistinguishable by 2ARE privacy.

Hybrid 2: Programming Signature Verification. For every $i \in H \setminus \{j\}$, replace the verification key and signature used by party j to authenticate C_i with

$$(vk_{j,i}\{C_i\}, \sigma_{j,i}\{C_i\}) \leftarrow \text{PSIG}.\widetilde{\text{Gen}}(1^\lambda, C_i).$$

The programmed verification keys replace the corresponding ordinary keys wherever they are hardwired into party j 's evaluation program, and the programmed signatures are used in the simulated 2ARE outputs. Programming indistinguishability gives

$$\text{Hyb}_1^b \approx_c \text{Hyb}_2^b.$$

We henceforth condition on all programmed verification keys generated in Hybrid 2 satisfying unique verification. The excluded choices have negligible probability and therefore contribute only a negligible amount to the distinguishing advantage.

Hybrid 3: Programming the Proof System. Replace the TNIZK string by one programmed at tag j :

$$(\tilde{\rho}, \tau) \leftarrow \text{TNIZK}.\widetilde{\text{Gen}}(1^\lambda, j).$$

Then replace party j 's honest proof by

$$\tilde{\pi}_j \leftarrow \text{TNIZK}.\widetilde{\text{P}}(\tilde{\rho}, \tau, j, (\eta, C_j, ct_j)).$$

Setup indistinguishability and ordinary zero knowledge give

$$\text{Hyb}_2^b \approx_c \text{Hyb}_3^b.$$

We henceforth condition on a programmed string $\tilde{\rho}$ (with its trapdoor τ) that satisfies extraction simultaneously for every tag different from j , statement, and accepting proof. The excluded choices contribute only a negligible amount to the distinguishing advantage.

Hybrid 4: Simulating Partial Decryption. In this hybrid, instead of obfuscating

$$\text{EV}_j[f, \tilde{\rho}, \eta, vk_{j,H \setminus \{j\}}\{C_{H \setminus \{j\}}\}, vk_{j,\overline{H}}, ct_j, sek_j, K_j],$$

we obfuscate

$$\text{EV}'_j[f, f_{H, \mathbf{x}_H^0}, \tilde{\rho}, \eta, vk_{j,H \setminus \{j\}}\{C_{H \setminus \{j\}}\}, vk_{j,\overline{H}}, ct_j, \textcolor{red}{sek_j}, \tau, K_j].$$

The program EV'_j extracts the other parties' plaintexts and keys, computes the output using the residual function, and invokes the MHE partial-decryption simulator. It is specified in Figure 5.

Figure 5: Partial-decryption-simulating program EV'_j

Hardwired:

- f , the circuit to be homomorphically evaluated;
- $f_{H, \mathbf{x}_H^0}$;
- $\tilde{\rho}, \eta$, the programmed verification keys $vk_{j,H \setminus \{j\}}\{C_{H \setminus \{j\}}\}$, and the ordinary keys $vk_{j,\overline{H}}$;
- party j 's ciphertext ct_j ;
- τ , the **TNIZK extraction trapdoor**; and

Figure 5 (continued)

- K_j .

Input: $C_{-j}, ct_{-j}, \sigma_{j,-j}, \pi_{-j}$.

Computation:

1. **Verify signatures and proofs:** Perform the same signature and TNIZK verification checks as EV_j and output $\perp$ if any check fails.
2. **Evaluate:** Let $ct := (ct_1, \dots, ct_m)$ and compute

$$ct^* := \text{MHE.Eval}_\eta(f, ct).$$

3. **For every $i \neq j$, extract**

$$(x_i, r_i, s_i) := \text{TNIZK.E}(\tilde{\rho}, \tau, i, (\eta, C_i, ct_i), \pi_i), \quad sek_i := G_{\text{sek}}(s_i).$$

4. **Let $x_{\bar{H}} := (x_i)_{i \in \bar{H}}$ and compute**

$$y := f_{H, x_H^0}(x_{\bar{H}}).$$

5. **Partially decrypt:** Derive

$$r_j := \text{PPRF.Eval}(K_j, (C_{-j}, ct_{-j}))$$

and output

$$d_j \leftarrow \text{MHE.PDec}_\eta(j, sek_{-j}, ct^*, y; r_j).$$

We show that this change is indistinguishable using the Nondeterministic pIO Lemma 2. Fix the programmed string, keys, and trapdoor subject to the unique signature verification and extraction assumptions above. Consider the relation

$$R_{\text{EV}}[vk_{j, H \setminus \{j\}}, \{C_{H \setminus \{j\}}\}, vk_{j, \bar{H}}, \tilde{\rho}]((C_{-j}, ct_{-j}); (\sigma_{j,-j}, \pi_{-j})),$$

whose instance is (C_{-j}, ct_{-j}) and whose witness is $(\sigma_{j,-j}, \pi_{-j})$. The relation accepts if and only if all signature and proof checks in EV_j and EV'_j accept. Let $L_{\text{EV}, \lambda}$ be the corresponding language. If the relation rejects, both programs output $\perp$.

We now show the required pointwise indistinguishability toward applying the Nondeterministic pIO Lemma 2. By our conditioning on $\tilde{\rho}$, extraction holds simultaneously for every accepting proof under a tag different from j . It therefore applies simultaneously to all accepting proofs in all instances of $L_{\text{EV}, \lambda}$.

For every accepted NIZK instance (η, C_i, ct_i) , extraction provides $x_i \in \{0, 1\}^n$ and $r_i, s_i \in \{0, 1\}^\lambda$ determining both values. Consequently,

$$|L_{\text{EV}, \lambda}| \leq 2^{(m-1)(n+2\lambda)}.$$

Fix an instance $(C_{-j}, ct_{-j}) \in L_{\text{EV}, \lambda}$. By the unique signature verification assumption stated after Hybrid 2, the commitments of the parties in $H \setminus \{j\}$ are fixed to their honestly generated

values. The extraction assumption gives an opening $C_i = \text{Com}(x_i; r_i)$ for every $i \neq j$. Perfect binding of the honest commitments therefore implies $x_i = x_i^b$ for every $i \in H \setminus \{j\}$. Furthermore, x_i is consistently encrypted by ct_i . Hence ct^* corresponds to

$$f_{H, x_H^b}(\mathbf{x}_H) = f_{H, x_H^0}(\mathbf{x}_H) = y.$$

Partial-decryption simulation of MHE implies distinguishing advantage at most $2^{-\tilde{\lambda}^\epsilon}$ and by our choice of $\tilde{\lambda}$,

$$2^{-\tilde{\lambda}^\epsilon} \leq \frac{2^{-(n+2\lambda)}}{|L_{\text{EV}, \lambda}|} \leq \frac{\text{negl}(\lambda)}{|L_{\text{EV}, \lambda}|}.$$

The iO and puncturable-PRF advantages satisfy the same bound. The Nondeterministic pIO Lemma therefore implies that the obfuscations of EV_j and EV'_j are computationally indistinguishable:

$$\text{Hyb}_3^b \approx_c \text{Hyb}_4^b.$$

Hybrid 5: Replacing Party j 's Ciphertext. Replace

$$ct_j := \text{MHE.Enc}_\eta(G_{\text{sek}}(s_j), x_j^b) \quad \text{by} \quad \begin{array}{l} \widetilde{\text{sek}}_j \leftarrow \{0, 1\}^{\tilde{\lambda}}, \\ \tilde{ct}_j := \text{MHE.Enc}_\eta(\widetilde{\text{sek}}_j, 0^n). \end{array}$$

By PRG security and MHE semantic security,

$$\text{Hyb}_4^b \approx_c \text{Hyb}_5^b.$$

Hybrid 6: Replacing Party j 's Commitment. Replace

$$C_j := \text{Com}(x_j^b; r_j) \quad \text{by} \quad \tilde{C}_j \leftarrow \text{Com}(0^n).$$

Computational hiding gives

$$\text{Hyb}_5^b \approx_c \text{Hyb}_6^b.$$

Conclusion. The distribution Hyb_6^b is independent of b : party j 's commitment commits to 0^n , its ciphertext encrypts 0^n , its proof is simulated, and its evaluation program depends on the honest inputs only through the common residual function f_{H, x_H^0} . Therefore

$$\text{Hyb}_6^0 \equiv \text{Hyb}_6^1.$$

Combining the hybrids proves neighboring-input robustness for every $H \subseteq [m]$.

Padding. All obfuscated programs in the construction of Figure 3 are padded to the size of the largest corresponding program appearing in the hybrids above. $\square$

Proposition 1 and Lemma 1 prove Theorem 1.

4 Setup-Free RARE for an Honest Majority

In this section we give a construction of a RARE without a common random string in the honest-majority setting.

Theorem 2 (restated). *Assuming iO and LWE, both with subexponential security, there exist setup-free RAREs for all efficiently computable functions in the honest-majority model.*

4.1 Construction

We now describe all components of the construction. Its security analysis will be given separately.

Ingredients. The construction uses the following primitives:

1. a computationally hiding, perfectly binding non-interactive commitment scheme Com for messages in $\{0, 1\}^n$, using randomness in $\{0, 1\}^\lambda$;
2. pseudorandom generators

$$G_\eta: \{0, 1\}^{\ell_\eta} \rightarrow \{0, 1\}^{\tilde{\lambda}}, \quad G_{\text{sek}}: \{0, 1\}^{\ell_{\text{sek}}} \rightarrow \{0, 1\}^{\tilde{\lambda}},$$

whose outputs are, respectively, an MHE common random string and an MHE secret key at security parameter $\tilde{\lambda}$. We require G_{sek} to be subexponentially secure and set $\ell_{\text{sek}}^\varepsilon \geq 2m\lambda$ where ε is the subexponential security constant. We also set $\ell_\eta = \lambda$. $\tilde{\lambda}$ is set below;

3. an (m, m) -MTNIZK MTNIZK satisfying ℓ -zero knowledge for $\ell \geq 2m\lambda$, for the relation

$$R := R_\eta \cup R_{\text{ct}},$$

where

$$R_\eta := \left\{ (\tilde{\eta}; s) : s \in \{0, 1\}^{\ell_\eta}, \quad \tilde{\eta} = G_\eta(s) \right\}$$

and

$$\begin{aligned} R_{\text{ct}} := & \left\{ ((\eta, C, ct); (x, r, s)) : \right. \\ & x \in \{0, 1\}^n, \quad r \in \{0, 1\}^\lambda, \quad s \in \{0, 1\}^{\ell_{\text{sek}}}, \\ & C = \text{Com}(x; r), \quad \text{sek} = G_{\text{sek}}(s), \quad ct = \text{MHE.Enc}_\eta(\text{sek}, x) \left. \right\}; \end{aligned}$$

4. a programmable signature scheme **PSIG**;
5. a 2ARE for the authentication function **Auth** defined below;
6. a subexponentially secure secret-key multi-key homomorphic encryption scheme **MHE** in the common random string model for a circuit class $\mathcal{C} \ni f$, where f is the multi-party function to be computed;
7. a subexponentially secure indistinguishability obfuscator **iO**; and
8. a subexponentially secure puncturable PRF **PPRF** whose range is long enough to supply the seeds and randomness used by the programs below.

For simplicity, G_{sek} , **MHE**, **iO**, and **PPRF** have a common subexponential security exponent $\varepsilon > 0$ (if their exponents differ, we take ε to be the smallest of the four). We choose the security parameter $\tilde{\lambda} = \tilde{\lambda}(\lambda)$ so that

$$\tilde{\lambda}^\varepsilon \geq m(n + 3\lambda + \ell_{\text{sek}}) \geq m(n + 3\lambda + (2m\lambda)^{1/\varepsilon}).$$

2ARE Authentication of Honest Parties. We use 2ARE to allow parties to authenticate one another's public values. The authenticated values are the MTNIZK strings and the input commitments. For $i \neq j$, $vk_{i,j}^\rho$ is the verification key used by party i to authenticate party j 's MTNIZK string ρ_j , and $vk_{i,j}^C$ is the verification key used to authenticate its commitment C_j . The corresponding signatures are $\sigma_{i,j}^\rho$ and $\sigma_{i,j}^C$. The mechanism is specified in Figure 6.

Figure 6: Authentication of honest parties using 2ARE

For each ordered pair of distinct parties (i, j) , party i independently generates two programmable-signature key pairs:

$$(ssk_{i,j}^\rho, vk_{i,j}^\rho) \leftarrow \text{PSIG.Gen}(1^\lambda), \quad (ssk_{i,j}^C, vk_{i,j}^C) \leftarrow \text{PSIG.Gen}(1^\lambda).$$

For every unordered pair $\{i, j\}$, define the two-party function

$$\begin{aligned} \text{Auth}_{i,j} &((ssk_{i,j}^\rho, ssk_{i,j}^C, \rho_i, C_i), \\ &\quad (ssk_{j,i}^\rho, ssk_{j,i}^C, \rho_j, C_j)) \\ &:= (\text{PSIG.Sig}(ssk_{i,j}^\rho, \rho_j), \text{PSIG.Sig}(ssk_{i,j}^C, C_j), \\ &\quad \text{PSIG.Sig}(ssk_{j,i}^\rho, \rho_i), \text{PSIG.Sig}(ssk_{j,i}^C, C_i)) \\ &=: (\sigma_{i,j}^\rho, \sigma_{i,j}^C, \sigma_{j,i}^\rho, \sigma_{j,i}^C). \end{aligned}$$

The parties run the two local encoders of a 2ARE for $\text{Auth}_{i,j}$. Party i uses local input

$$u_{i,j} := (ssk_{i,j}^\rho, ssk_{i,j}^C, \rho_i, C_i)$$

and obtains

$$\hat{g}_{i,j} \leftarrow \text{2ARE.Enc}_{i,j}^{(i)}(u_{i,j});$$

party j acts symmetrically to obtain $\hat{g}_{j,i}$. The pairwise decoder recovers the four signatures above from

$$\hat{g}_{i,j} + \hat{g}_{j,i}.$$

The group of the resulting RARE is the product of the 2ARE groups over the $\binom{m}{2}$ unordered pairs. Party i 's group component $\hat{g}_i$ places $\hat{g}_{i,j}$ in coordinate $\{i, j\}$ for every $j \neq i$ and the identity element in all other coordinates. Consequently,

$$\left(\sum_{i \in [m]} \hat{g}_i \right)_{\{i,j\}} = \hat{g}_{i,j} + \hat{g}_{j,i} \quad \text{for every } \{i, j\}.$$

Obfuscated Programs. For a fixed party i , write

$$\begin{aligned} vk_{i,-i}^\rho &:= (vk_{i,j}^\rho)_{j \neq i}, & vk_{i,-i}^C &:= (vk_{i,j}^C)_{j \neq i}, \\ \sigma_{i,-i}^\rho &:= (\sigma_{i,j}^\rho)_{j \neq i}, & \sigma_{i,-i}^C &:= (\sigma_{i,j}^C)_{j \neq i}, \\ \rho_{-i} &:= (\rho_j)_{j \neq i}, & \tilde{\eta}_{-i} &:= (\tilde{\eta}_j)_{j \neq i}, \\ C_{-i} &:= (C_j)_{j \neq i}, & ct_{-i} &:= (ct_j)_{j \neq i}, \\ \pi_{-i}^\eta &:= (\pi_j^\eta)_{j \neq i}, & \pi_{-i}^{ct} &:= (\pi_j^{ct})_{j \neq i}. \end{aligned}$$

The first obfuscated program proves that party i 's MHE-string share is pseudorandom. We denote it by

$$\text{MSSP}_i[vk_{i,-i}^\rho, \rho_i, \tilde{\eta}_i, s_i^\eta, K_i^\eta](\rho_{-i}, \sigma_{i,-i}^\rho).$$

Its operation is specified in Figure 7.

Figure 7: MHE-string-share proof program MSSP_i

Hardwired:

- $\mathbf{vk}_{i,-i}^\rho$, the verification keys used by party i to authenticate the other parties' MTNIZK strings;
- ρ_i , party i 's MTNIZK string;
- $s_i^\eta \in \{0, 1\}^{\ell_\eta}$ and $\tilde{\eta}_i := G_\eta(s_i^\eta)$; and
- K_i^η , a puncturable-PRF key.

Input:

- ρ_{-i} , the other parties' MTNIZK strings; and
- $\sigma_{i,-i}^\rho$, their corresponding signatures.

Computation:

1. **Verify signatures:** For every $j \neq i$, check

$$\text{PSIG.Ver}(\mathbf{vk}_{i,j}^\rho, \rho_j, \sigma_{i,j}^\rho) = 1.$$

If any check fails, output $\perp$.

2. **Generate prover randomness:** Compute

$$r_i^\eta := \text{PPRF.Eval}(K_i^\eta, \rho_{-i}).$$

3. **Prove:** Let $\rho := (\rho_1, \dots, \rho_m)$. Using tag i , output a proof that $\tilde{\eta}_i$ is pseudorandom:

$$\pi_i^\eta := \text{MTNIZK.P}(\rho, i, \tilde{\eta}_i, s_i^\eta; r_i^\eta).$$

The second obfuscated program creates party i 's MHE ciphertext and proves that it encrypts the value committed in C_i under the aggregate MHE string. We denote it by

$$\text{CTP}_i[\mathbf{vk}_{i,-i}^\rho, \rho_i, \tilde{\eta}_i, C_i, x_i, r_i, K_i^{\text{ct}}](\rho_{-i}, \sigma_{i,-i}^\rho, \tilde{\eta}_{-i}, \pi_{-i}^\eta).$$

Its operation is specified in Figure 8.

Figure 8: Ciphertext-and-proof program CTP_i

Hardwired:

- $\mathbf{vk}_{i,-i}^\rho$ and ρ_i ;
- $\tilde{\eta}_i$, party i 's MHE-string share;
- x_i , party i 's input, and commitment randomness r_i such that $C_i = \text{Com}(x_i; r_i)$; and
- K_i^{ct} , a puncturable-PRF key.

Input:

Figure 8 (continued)

- ρ_{-i} and $\sigma_{i,-i}^\rho$;
- $\tilde{\eta}_{-i}$, the other parties' MHE-string shares; and
- π_{-i}^η , the corresponding proofs of pseudorandomness.

Computation:

1. **Verify signatures:** For every $j \neq i$, check

$$\text{PSIG.Ver}(vk_{i,j}^\rho, \rho_j, \sigma_{i,j}^\rho) = 1.$$

If any check fails, output $\perp$.

2. **Verify MHE-string-share proofs:** Let $\rho := (\rho_1, \dots, \rho_m)$. For every $j \neq i$, check

$$\text{MTNIZK.V}(\rho, j, \tilde{\eta}_j, \pi_j^\eta) = 1.$$

If any check fails, output $\perp$.

3. **Aggregate the MHE string:** Compute

$$\eta := \bigoplus_{j \in [m]} \tilde{\eta}_j.$$

4. **Derive the encryption key and prover randomness:** Compute

$$s_i^{\text{sek}} \| r_i^{\text{ct}} := \text{PPRF.Eval}(K_i^{\text{ct}}, (\rho_{-i}, \tilde{\eta}_{-i})), \quad \text{sek}_i := G_{\text{sek}}(s_i^{\text{sek}}).$$

5. **Encrypt and prove:** Compute

$$ct_i := \text{MHE.Enc}_\eta(\text{sek}_i, x_i)$$

and then, using tag i , compute a proof that ct_i is a valid ciphertext consistent with C_i :

$$\pi_i^{\text{ct}} := \text{MTNIZK.P}(\rho, i, (\eta, C_i, ct_i), (x_i, r_i, s_i^{\text{sek}}); r_i^{\text{ct}}).$$

Output $(ct_i, \pi_i^{\text{ct}})$.

The third obfuscated program verifies the other parties' commitments, MHE-string shares, and ciphertexts, homomorphically evaluates f , and produces i 's partial decryption. We denote it by

$$\begin{aligned} \text{EV}_i[f, vk_{i,-i}^\rho, \rho_i, vk_{i,-i}^C, C_i, \\ \tilde{\eta}_i, x_i, K_i^{\text{ct}}, K_i^{\text{EV}}](\rho_{-i}, \sigma_{i,-i}^\rho, C_{-i}, \sigma_{i,-i}^C, \\ \tilde{\eta}_{-i}, \pi_{-i}^\eta, ct_{-i}, \pi_{-i}^{\text{ct}}). \end{aligned}$$

Its operation is specified in Figure 9.

Figure 9: Homomorphic-evaluation program EV_i

Hardwired:

- f , the circuit to be evaluated;
- $\mathbf{vk}_{i,-i}^\rho$, ρ_i , $\mathbf{vk}_{i,-i}^C$, and C_i ;
- $\tilde{\eta}_i$, party i 's MHE-string share;
- x_i , party i 's input;
- K_i^{ct} , a puncturable-PRF key (also hardwired into CTP_i); and
- K_i^{EV} , a puncturable-PRF key used to derive partial-decryption randomness.

Input:

- ρ_{-i} and $\sigma_{i,-i}^\rho$;
- C_{-i} and $\sigma_{i,-i}^C$;
- $\tilde{\eta}_{-i}$ and π_{-i}^η ; and
- ct_{-i} and π_{-i}^{ct} .

Computation:

1. **Verify signatures:** For every $j \neq i$, check

$$\text{PSIG.Ver}(\mathbf{vk}_{i,j}^\rho, \rho_j, \sigma_{i,j}^\rho) = 1 \quad \text{and} \quad \text{PSIG.Ver}(\mathbf{vk}_{i,j}^C, C_j, \sigma_{i,j}^C) = 1.$$

If any check fails, output $\perp$.

2. **Verify MHE-string-share proofs:** Let $\rho := (\rho_1, \dots, \rho_m)$. For every $j \neq i$, check

$$\text{MTNIZK.V}(\rho, j, \tilde{\eta}_j, \pi_j^\eta) = 1.$$

If any check fails, output $\perp$.

3. **Aggregate the MHE string:** Compute

$$\eta := \bigoplus_{j \in [m]} \tilde{\eta}_j.$$

4. **Verify ciphertext proofs:** For every $j \neq i$, check

$$\text{MTNIZK.V}(\rho, j, (\eta, C_j, ct_j), \pi_j^{ct}) = 1.$$

If any check fails, output $\perp$.

5. **Recover party i 's encryption key and ciphertext:** Compute

$$s_i^{\text{sek}} \| r_i^{ct} := \text{PPRF.Eval}(K_i^{\text{ct}}, (\rho_{-i}, \tilde{\eta}_{-i})), \quad \text{sek}_i := G_{\text{sek}}(s_i^{\text{sek}}),$$

and

$$ct_i := \text{MHE.Enc}_\eta(\text{sek}_i, x_i).$$

Figure 9 (continued)

6. **Evaluate:** Let $\mathbf{ct} := (ct_1, \dots, ct_m)$ and compute

$$ct^* := \text{MHE.Eval}_\eta(f, \mathbf{ct}).$$

7. **Partially decrypt:** Derive

$$r_i^{\text{dec}} := \text{PPRF.Eval}(K_i^{\text{EV}}, (\rho_{-i}, \mathbf{C}_{-i}, \tilde{\eta}_{-i}, \mathbf{ct}_{-i})),$$

and output

$$d_i \leftarrow \text{MHE.PDec}_\eta(\text{sek}_i, ct^*; r_i^{\text{dec}}).$$

The complete RARE encoder and decoder are given in Figures 10 and 11, respectively.

Figure 10: RARE encoder

To encode party i 's input x_i :

1. Commit to the input: sample commitment randomness r_i and set

$$C_i := \text{Com}(x_i; r_i),$$

2. Generate party i 's MTNIZK string and its pseudorandom MHE-string share:

$$\rho_i \leftarrow \{0, 1\}^\lambda, \quad s_i^\eta \leftarrow \{0, 1\}^{\ell_\eta}, \quad \tilde{\eta}_i := G_\eta(s_i^\eta).$$

3. Generate puncturable-PRF keys

$$K_i^\eta \leftarrow \{0, 1\}^{\tilde{\lambda}}, \quad K_i^{\text{ct}} \leftarrow \{0, 1\}^{\tilde{\lambda}}, \quad K_i^{\text{EV}} \leftarrow \{0, 1\}^{\tilde{\lambda}}.$$

4. For every $j \neq i$, sample signing and verification keys

$$(ssk_{i,j}^\rho, vk_{i,j}^\rho) \leftarrow \text{PSIG.Gen}(1^\lambda), \quad (ssk_{i,j}^C, vk_{i,j}^C) \leftarrow \text{PSIG.Gen}(1^\lambda).$$

5. Compute (padded) obfuscations

$$\widetilde{\text{MSSP}}_i \leftarrow \text{iO} \left(\text{MSSP}_i[\mathbf{vk}_{i,-i}^\rho, \rho_i, \tilde{\eta}_i, s_i^\eta, K_i^\eta] \right),$$

$$\widetilde{\text{CTP}}_i \leftarrow \text{iO} \left(\text{CTP}_i[\mathbf{vk}_{i,-i}^\rho, \rho_i, \tilde{\eta}_i, C_i, x_i, r_i, K_i^{\text{ct}}] \right),$$

and

$$\widetilde{\text{EV}}_i \leftarrow \text{iO} \left(\text{EV}_i[f, \mathbf{vk}_{i,-i}^\rho, \rho_i, \mathbf{vk}_{i,-i}^C, C_i, \tilde{\eta}_i, x_i, K_i^{\text{ct}}, K_i^{\text{EV}}] \right).$$

6. For every $j \neq i$, run party i 's local encoder in the 2ARE for $\text{Auth}_{i,j}$ on

$$u_{i,j} := (ssk_{i,j}^\rho, ssk_{i,j}^C, \rho_i, C_i)$$

to obtain

$$\hat{g}_{i,j} \leftarrow \text{2ARE.Enc}_{i,j}^{(i)}(u_{i,j}).$$

Let $\hat{g}_i$ be the corresponding product-group element.

Figure 10 (continued)

7. The non-interactive component is

$$\widehat{z}_i := (C_i, \rho_i, \widetilde{\eta}_i, \widetilde{\text{MSSP}}_i, \widetilde{\text{CTP}}_i, \widetilde{\text{EV}}_i),$$

and the local encoding is $(\widehat{z}_i, \widehat{g}_i)$.

Figure 11: RARE decoder

The evaluator is given the non-interactive components, parsed as

$$\widehat{z} = (C, \rho, \widetilde{\eta}, \widetilde{\text{MSSP}}, \widetilde{\text{CTP}}, \widetilde{\text{EV}}),$$

together with the aggregate product-group element $\widehat{g} := \sum_{i \in [m]} \widehat{g}_i$. It proceeds as follows:

1. **Decode the 2AREs:** For every unordered pair $\{i, j\}$, compute

$$(\sigma_{i,j}^\rho, \sigma_{i,j}^C, \sigma_{j,i}^\rho, \sigma_{j,i}^C) := \text{2ARE.Dec}_{i,j}(\widehat{g}_{\{i,j\}}).$$

2. **Generate MHE-string-share proofs:** For every $i \in [m]$, compute

$$\pi_i^\eta := \widetilde{\text{MSSP}}_i(\rho_{-i}, \sigma_{i,-i}^\rho).$$

3. **Generate ciphertexts and ciphertext proofs:** For every $i \in [m]$, compute

$$(ct_i, \pi_i^{ct}) := \widetilde{\text{CTP}}_i(\rho_{-i}, \sigma_{i,-i}^\rho, \widetilde{\eta}_{-i}, \pi_{-i}^\eta).$$

4. **Generate partial decryptions:** For every $i \in [m]$, compute

$$d_i \leftarrow \widetilde{\text{EV}}_i(\rho_{-i}, \sigma_{i,-i}^\rho, C_{-i}, \sigma_{i,-i}^C, \widetilde{\eta}_{-i}, \pi_{-i}^\eta, ct_{-i}, \pi_{-i}^{ct}).$$

5. **Decrypt:** Let

$$\eta := \bigoplus_{j \in [m]} \widetilde{\eta}_j, \quad \mathbf{d} := (d_1, \dots, d_m),$$

and output

$$y := \text{MHE.Dec}_\eta(\mathbf{d}).$$

Correctness. When all parties are honest, every pairwise 2ARE decodes to the corresponding signatures, so all signature checks pass. Each $\widetilde{\text{MSSP}}_i$ produces an accepting proof for the honestly generated share $\widetilde{\eta}_i = G_\eta(s_i^\eta)$. Hence every $\widetilde{\text{CTP}}_i$ uses the same aggregate string $\eta = \bigoplus_j \widetilde{\eta}_j$, derives its secret key consistently with $\widetilde{\text{EV}}_i$, and produces a ciphertext and accepting proof for the value committed in C_i . The evaluation programs therefore evaluate f on encryptions of $(x_1, \dots, x_m)$ under the same MHE common random string and output the corresponding partial decryptions. Perfect correctness of MHE implies that the decoder outputs $f(x_1, \dots, x_m)$ with probability one.

4.2 Security Analysis

Proposition 2. *The RARE construction above satisfies neighboring-input $\mathcal{H}_{\text{maj}}$ -robustness.*

Proof. Fix $H \in \mathcal{H}_{\text{maj}}$ and neighboring input vectors $\mathbf{x}_H^0 = (x_i^0)_{i \in H}$ and $\mathbf{x}_H^1 = (x_i^1)_{i \in H}$ such that

$$f_{H, \mathbf{x}_H^0} \equiv f_{H, \mathbf{x}_H^1}.$$

Let $j \in H$ be the unique party on whose input the two vectors may differ; thus, $x_i^0 = x_i^1$ for every $i \in H \setminus \{j\}$.

For an honest input vector $\mathbf{u}_H = (u_i)_{i \in H}$, define the distribution of the evaluator's view by

$$\text{View}(H, \mathbf{u}_H) := (\widehat{\mathbf{z}}_H, \widehat{\mathbf{g}}_H), \quad \widehat{\mathbf{g}}_H := \sum_{i \in H} \widehat{g}_i,$$

where, independently for every $i \in H$,

$$(\widehat{z}_i, \widehat{g}_i) \leftarrow \text{Enc}(u_i, i).$$

Equivalently, $\text{View}(H, \mathbf{u}_H)$ consists of all non-interactive components produced by the parties in H , together with the sum of their group components. Our goal is to prove

$$\text{View}(H, \mathbf{x}_H^0) \approx_c \text{View}(H, \mathbf{x}_H^1)$$

We establish this claim through a sequence of hybrid distributions. In the description of each hybrid, components changed relative to the preceding hybrid will be colored **red**.

For each $b \in \{0, 1\}$, define hybrids $\text{Hyb}_0^b, \dots, \text{Hyb}_9^b$. We will show that each consecutive pair is computationally indistinguishable and that the final distribution is independent of b .

Hybrid 0: Real. This is the real distribution

$$\text{Hyb}_0^b := \text{View}(H, \mathbf{x}_H^b).$$

Hybrid 1: Simulating Non-Corrupted 2ARE. For every unordered pair $\{\alpha, \beta\} \subseteq H$, let

$$\boldsymbol{\sigma}_{\{\alpha, \beta\}} := (\sigma_{\alpha, \beta}^\rho, \sigma_{\alpha, \beta}^C, \sigma_{\beta, \alpha}^\rho, \sigma_{\beta, \alpha}^C)$$

be the output of the corresponding authentication function. In coordinate $\{\alpha, \beta\}$ of the aggregate group element, replace

$$\widehat{g}_{\alpha, \beta} + \widehat{g}_{\beta, \alpha} \quad \text{by} \quad \widetilde{g}_{\{\alpha, \beta\}} \leftarrow \text{2ARE.Sim}(\boldsymbol{\sigma}_{\{\alpha, \beta\}}).$$

The distributions Hyb_0^b and Hyb_1^b are computationally indistinguishable by 2ARE privacy.

Hybrid 2: Programming Signature Verification. For every $i \in H \setminus \{j\}$, replace the verification key and signature used by party j to authenticate ρ_i with

$$(vk_{j,i}^\rho, \sigma_{j,i}^\rho) \leftarrow \text{PSIG.Gen}(1^\lambda, \rho_i).$$

Similarly, replace the pair used by party j to authenticate C_i with

$$(vk_{j,i}^C, \sigma_{j,i}^C) \leftarrow \text{PSIG.Gen}(1^\lambda, C_i).$$

The programmed verification keys replace the corresponding ordinary keys wherever they are hard-wired into party j 's three obfuscated programs, and the programmed signatures are used in the simulated 2ARE outputs. Programming indistinguishability gives

$$\text{Hyb}_1^b \approx_c \text{Hyb}_2^b.$$

Hybrid 3: Programming Honest Reference Strings. Replace ρ_H with strings programmed at tag j :

$$(\tilde{\rho}_H, \tau) \leftarrow \text{MTNIZK}.\widetilde{\text{Gen}}(1^\lambda, j, H).$$

Setup indistinguishability of the MTNIZK gives

$$\text{Hyb}_2^b \approx_c \text{Hyb}_3^b.$$

We henceforth condition on programmed strings $\tilde{\rho}_H$ (with their trapdoor τ) that satisfy extraction simultaneously for every choice of the remaining reference strings, every tag different from j , statement, and accepting proof. We also assume that all programmed verification keys from Hybrid 2 satisfy unique verification. The excluded choices contribute only a negligible amount to the distinguishing advantage.

Hybrid 4: Simulating the MHE-String-Share Proof Program. In this hybrid, instead of obfuscating

$$\text{MSSP}_j \left[\mathbf{vk}_{j,H \setminus \{j\}}^\rho \{\tilde{\rho}_{H \setminus \{j\}}\}, \mathbf{vk}_{j,\overline{H}}^\rho, \tilde{\rho}_j, \tilde{\eta}_j, s_j^\eta, K_j^\eta \right],$$

we obfuscate

$$\text{MSSP}'_j \left[\mathbf{vk}_{j,H \setminus \{j\}}^\rho \{\tilde{\rho}_{H \setminus \{j\}}\}, \mathbf{vk}_{j,\overline{H}}^\rho, \tilde{\rho}_j, \tilde{\eta}_j, \textcolor{red}{s_j^\eta}, \textcolor{red}{\tau}, K_j^\eta \right].$$

The program MSSP'_j has the trapdoor τ hardwired instead of s_j^η and simulates the proof for $\tilde{\eta}_j$. It is specified in Figure 12.

Figure 12: Simulated MHE-string-share proof program MSSP'_j

Hardwired:

- the programmed verification keys $\mathbf{vk}_{j,H \setminus \{j\}}^\rho \{\tilde{\rho}_{H \setminus \{j\}}\}$ and the ordinary keys $\mathbf{vk}_{j,\overline{H}}^\rho$;
- $\tilde{\rho}_j$ and $\tilde{\eta}_j$;
- $\textcolor{red}{\tau}$, the MTNIZK simulation trapdoor; and
- K_j^η .

Input: ρ_{-j} and $\sigma_{j,-j}^\rho$.

Computation:

1. **Verify signatures:** For every $i \neq j$, check

$$\text{PSIG.Ver}(\mathbf{vk}_{j,i}^\rho, \rho_i, \sigma_{j,i}^\rho) = 1.$$

If any check fails, output $\perp$.

2. **Generate prover randomness:** Compute

$$r_j^\eta := \text{PPRF.Eval}(K_j^\eta, \rho_{-j}).$$

3. **Prove:** Let $\tilde{\rho}$ be determined by $\tilde{\rho}_j$ and ρ_{-j} . Using tag j , output a simulated proof for $\tilde{\eta}_j$:

$$\textcolor{red}{\pi_j^\eta} := \text{MTNIZK}.\widetilde{\text{P}}(\tilde{\rho}, \textcolor{red}{\tau}, j, \tilde{\eta}_j; r_j^\eta).$$

We show that this change is indistinguishable using the Nondeterministic pIO Lemma 2. Consider the NP relation

$$R_{\text{MSSP}} \left[\mathbf{vk}_{j,H \setminus \{j\}}^\rho \{ \tilde{\rho}_{H \setminus \{j\}} \}, \mathbf{vk}_{j,\overline{H}}^\rho \right] (\rho_{-j}, \sigma_{j,-j}^\rho).$$

The relation hardwires the signature-verification keys used by MSSP_j and MSSP'_j . Its instance is ρ_{-j} and its witness is $\sigma_{j,-j}^\rho$. It accepts precisely when

$$\begin{aligned} \text{PSIG.Ver}(vk_{j,i}^\rho \{ \tilde{\rho}_i \}, \rho_i, \sigma_{j,i}^\rho) &= 1 \quad \text{for every } i \in H \setminus \{j\}, \\ \text{PSIG.Ver}(vk_{j,i}^\rho, \rho_i, \sigma_{j,i}^\rho) &= 1 \quad \text{for every } i \in \overline{H}. \end{aligned}$$

Let $L_{\text{MSSP},\lambda}$ be the corresponding language of instances ρ_{-j} for which such a signature witness exists.

We now show the required pointwise indistinguishability toward applying the Nondeterministic pIO Lemma 2. By the assumed unique verification, every instance $\rho_{-j} \in L_{\text{MSSP},\lambda}$ satisfies

$$\rho_{H \setminus \{j\}} = \tilde{\rho}_{H \setminus \{j\}}.$$

Thus, together with the hardwired $\tilde{\rho}_j$, every such instance determines an aggregate reference string whose coordinates in H are those programmed at tag j . Moreover,

$$|L_{\text{MSSP},\lambda}| \leq 2^{|\rho_{\overline{H}}|} \leq 2^{(m-1)\lambda}.$$

For every instance $\rho_{-j} \in L_{\text{MSSP},\lambda}$, the ℓ -zero-knowledge property gives

$$\text{MTNIZK.P}(\tilde{\rho}, j, \tilde{\eta}_j, s_j^\eta) \approx_{2^{-\ell} \cdot \text{negl}(\lambda)} \text{MTNIZK}.\tilde{\text{P}}(\tilde{\rho}, \tau, j, \tilde{\eta}_j).$$

By the choice $\ell \geq 2m\lambda$ and the bound above, this advantage is at most

$$\frac{\text{negl}(\lambda)}{|L_{\text{MSSP},\lambda}|}.$$

By the choice of $\tilde{\lambda}$, the advantages of iO and PPRF are at most

$$2^{-\tilde{\lambda}^\epsilon} < \frac{\text{negl}(\lambda)}{|L_{\text{MSSP},\lambda}|}.$$

Hence we can apply the Nondeterministic pIO Lemma 2, deducing that the obfuscations of MSSP_j and MSSP'_j are computationally indistinguishable, and therefore

$$\text{Hyb}_3^b \approx_c \text{Hyb}_4^b.$$

Hybrid 5: Replacing Party j 's MHE-String Share. Replace

$$\tilde{\eta}_j := G_\eta(s_j^\eta) \quad \text{by} \quad \eta_j \leftarrow \{0, 1\}^{\tilde{\lambda}},$$

and denote the uniform share by η_j from this hybrid onward. The seed s_j^η no longer appears after Hybrid 4. Thus, pseudorandomness of G_η gives

$$\text{Hyb}_4^b \approx_c \text{Hyb}_5^b.$$

Hybrid 6: Simulating Partial Decryption. In this hybrid, instead of obfuscating

$$\text{EV}_j[f, \mathbf{vk}_{j,H \setminus \{j\}}^\rho \{\tilde{\rho}_{H \setminus \{j\}}\}, \mathbf{vk}_{j,\overline{H}}^\rho, \tilde{\rho}_j, \\ \mathbf{vk}_{j,H \setminus \{j\}}^C \{C_{H \setminus \{j\}}\}, \mathbf{vk}_{j,\overline{H}}^C, C_j, \eta_j, x_j^b, K_j^{\text{ct}}, K_j^{\text{EV}}],$$

we obfuscate

$$\text{EV}'_j[f, \textcolor{red}{f}_{H, \mathbf{x}_H^0}, \mathbf{vk}_{j,H \setminus \{j\}}^\rho \{\tilde{\rho}_{H \setminus \{j\}}\}, \mathbf{vk}_{j,\overline{H}}^\rho, \tilde{\rho}_j, \\ \mathbf{vk}_{j,H \setminus \{j\}}^C \{C_{H \setminus \{j\}}\}, \mathbf{vk}_{j,\overline{H}}^C, C_j, \eta_j, x_j^b, K_j^{\text{ct}}, \textcolor{red}{\tau}, K_j^{\text{EV}}].$$

The program EV'_j changes the partial-decryption computation: it extracts the other parties' keys, computes the output using the residual function, and invokes the MHE partial-decryption simulator. The program is specified in Figure 13.

Figure 13: Partial-decryption-simulating program EV'_j

Hardwired:

- f , the circuit to be homomorphically evaluated;
- $\textcolor{red}{f}_{H, \mathbf{x}_H^0}$;
- the programmed reference-string and commitment verification keys for $H \setminus \{j\}$, and the ordinary keys for $\overline{H}$;
- $\tilde{\rho}_j$, C_j , and η_j ;
- x_j^b and K_j^{ct} ;
- $\textcolor{red}{\tau}$, the MTNIZK extraction trapdoor; and
- K_j^{EV} .

Input: ρ_{-j} , $\sigma_{j,-j}^\rho$, C_{-j} , $\sigma_{j,-j}^C$, $\tilde{\eta}_{-j}$, π_{-j}^η , ct_{-j} , and π_{-j}^{ct} .

Computation:

1. **Verify signatures:** For every $i \neq j$, check

$$\text{PSIG.Ver}(vk_{j,i}^\rho, \rho_i, \sigma_{j,i}^\rho) = 1 \quad \text{and} \quad \text{PSIG.Ver}(vk_{j,i}^C, C_i, \sigma_{j,i}^C) = 1.$$

If any check fails, output $\perp$.

2. **Verify MHE-string-share proofs:** Let $\tilde{\rho}$ be determined by $\tilde{\rho}_j$ and ρ_{-j} . For every $i \neq j$, check

$$\text{MTNIZK.V}(\tilde{\rho}, i, \tilde{\eta}_i, \pi_i^\eta) = 1.$$

If any check fails, output $\perp$.

3. **Aggregate the MHE string:** Compute

$$\eta := \eta_j \bigoplus_{i \in [m] \setminus \{j\}} \tilde{\eta}_i.$$

Figure 13 (continued)

4. **Verify ciphertext proofs:** For every $i \neq j$, check

$$\text{MTNIZK.V}(\tilde{\rho}, i, (\eta, C_i, ct_i), \pi_i^{ct}) = 1.$$

If any check fails, output $\perp$.

5. **Recover party j 's encryption key and ciphertext:** Compute

$$s_j^{\text{sek}} \| r_j^{ct} := \text{PPRF.Eval}(K_j^{ct}, (\rho_{-j}, \tilde{\eta}_{-j})), \quad \text{sek}_j := G_{\text{sek}}(s_j^{\text{sek}}),$$

and

$$ct_j := \text{MHE.Enc}_{\eta}(\text{sek}_j, x_j^b).$$

6. **Evaluate:** Let $ct := (ct_1, \dots, ct_m)$ and compute

$$ct^* := \text{MHE.Eval}_{\eta}(f, ct).$$

7. **For every $i \neq j$, extract**

$$(x_i, r_i, s_i^{\text{sek}}) := \text{MTNIZK.E}(\tilde{\rho}, \tau, i, (\eta, C_i, ct_i), \pi_i^{ct}),$$

$$\text{sek}_i := G_{\text{sek}}(s_i^{\text{sek}}).$$

8. **Let $x_{\bar{H}} := (x_i)_{i \in \bar{H}}$ and compute**

$$y := f_{H, x_H^0}(x_{\bar{H}}).$$

9. **Derive**

$$r_j^{\text{dec}} := \text{PPRF.Eval}(K_j^{\text{EV}}, (\rho_{-j}, C_{-j}, \tilde{\eta}_{-j}, ct_{-j})),$$

and output

$$d_j \leftarrow \text{MHE.PDec}_{\eta}(j, \text{sek}_{-j}, ct^*, y; r_j^{\text{dec}}).$$

We show that this change is indistinguishable using the Nondeterministic pIO Lemma 2. Fix the programmed strings, keys, signatures, and trapdoor subject to the unique signature verification and extraction assumptions above. Consider the relation

$$R_{\text{EV}}[\mathbf{vk}_{j, H \setminus \{j\}}^{\rho}, \{\tilde{\rho}_{H \setminus \{j\}}\}, \mathbf{vk}_{j, \bar{H}}^{\rho}, \tilde{\rho}_j, \mathbf{vk}_{j, H \setminus \{j\}}^C, \{C_{H \setminus \{j\}}\}, \mathbf{vk}_{j, \bar{H}}^C, \eta_j]$$

$$((\rho_{-j}, C_{-j}, \tilde{\eta}_{-j}, ct_{-j}),$$

$$(\sigma_{j, -j}^{\rho}, \sigma_{j, -j}^C, \pi_{-j}^{\eta}, \pi_{-j}^{ct})).$$

The instance is $(\rho_{-j}, C_{-j}, \tilde{\eta}_{-j}, ct_{-j})$, and the witness consists of the corresponding signatures and proofs. The relation accepts if and only if all signature, MHE-string-share-proof, and ciphertext-proof checks in the two programs accept. Let $L_{\text{EV}, \lambda}$ be the corresponding language. If the relation rejects, both programs output $\perp$.

We now show the required pointwise indistinguishability toward applying the Nondeterministic

pIO Lemma 2. By our conditioning on $\tilde{\rho}_H$, extraction holds simultaneously for every choice of $\rho_{\overline{H}}$ and every accepting proof under a tag different from j . It therefore applies simultaneously to all accepting proofs in all instances of $L_{\text{EV},\lambda}$.

We bound the size of this language by counting only values that can occur in accepted instances. There are at most $2^{(m-1)\lambda}$ possible vectors ρ_{-j} . Extraction from every accepting MHE-string-share proof gives a seed $s_i^\eta \in \{0,1\}^{\ell_\eta}$ satisfying $\tilde{\eta}_i = G_\eta(s_i^\eta)$, so there are at most $2^{(m-1)\ell_\eta} = 2^{(m-1)\lambda}$ possible accepted vectors $\tilde{\eta}_{-j}$. Extraction from every accepting ciphertext proof gives

$$x_i \in \{0,1\}^n, \quad r_i \in \{0,1\}^\lambda, \quad s_i^{\text{sek}} \in \{0,1\}^{\ell_{\text{sek}}},$$

which determine both $C_i = \text{Com}(x_i; r_i)$ and

$$ct_i = \text{MHE.Enc}_\eta(G_{\text{sek}}(s_i^{\text{sek}}), x_i).$$

Thus, there are at most $2^{(m-1)(n+\lambda+\ell_{\text{sek}})}$ possible accepted commitment-ciphertext vectors. Consequently,

$$|L_{\text{EV},\lambda}| \leq 2^{(m-1)(n+3\lambda+\ell_{\text{sek}})}.$$

For every instance

$$(\rho_{-j}, C_{-j}, \tilde{\eta}_{-j}, ct_{-j}) \in L_{\text{EV},\lambda},$$

unique verification fixes ρ_i and C_i for every $i \in H \setminus \{j\}$. The coordinates in H of the aggregate reference string are therefore those programmed at tag j . Extraction from every accepting ciphertext proof with tag $i \neq j$ gives

$$C_i = \text{Com}(x_i; r_i), \quad ct_i = \text{MHE.Enc}_\eta(G_{\text{sek}}(s_i^{\text{sek}}), x_i)$$

for every $i \neq j$. Perfect binding of the commitments implies $x_i = x_i^b$ for every $i \in H \setminus \{j\}$. Party j 's ciphertext is generated as in the real program and encrypts x_j^b . In both programs, let

$$ct^* := \text{MHE.Eval}_\eta(f, ct).$$

The ciphertext vector and the circuit f are identical in the two programs, so they compute the same ct^* . This evaluated ciphertext encrypts

$$y := f_{H, x_H^b}(x_{\overline{H}}) = f_{H, x_H^0}(x_{\overline{H}}).$$

Computational simulation of partial decryption then gives that

$$\text{MHE.PDec}_\eta(sek_j, ct^*) \text{ and } \widetilde{\text{MHE.PDec}}_\eta(j, sek_{-j}, ct^*, y)$$

are $2^{-\tilde{\lambda}^\varepsilon}$ -indistinguishable.

By our choice of $\tilde{\lambda}$, for every instance in $L_{\text{EV},\lambda}$, the outputs of EV_j and EV'_j are pointwise computationally indistinguishable with advantage at most

$$2^{-\tilde{\lambda}^\varepsilon} < 2^{-m(n+3\lambda+\ell_{\text{sek}})} \leq \frac{2^{-(n+3\lambda+\ell_{\text{sek}})}}{|L_{\text{EV},\lambda}|} \leq \frac{\text{negl}(\lambda)}{|L_{\text{EV},\lambda}|}.$$

Our choice of $\tilde{\lambda}$ also makes the advantages of iO and PPRF at most

$$2^{-\tilde{\lambda}^\varepsilon} < \frac{\text{negl}(\lambda)}{|L_{\text{EV},\lambda}|}.$$

Hence we can apply the Nondeterministic pIO Lemma 2, deducing that the obfuscations of EV_j and EV'_j are computationally indistinguishable. Consequently,

$$\text{Hyb}_5^b \approx_c \text{Hyb}_6^b.$$

Hybrid 7: Replacing Direct Ciphertext Generation. In this hybrid, instead of obfuscating

$$\text{EV}'_j[f, f_{H, \mathbf{x}_H^0}, \mathbf{vk}_{j, H \setminus \{j\}}^\rho \{\tilde{\rho}_{H \setminus \{j\}}\}, \mathbf{vk}_{j, \overline{H}}^\rho, \tilde{\rho}_j, \\ \mathbf{vk}_{j, H \setminus \{j\}}^C \{C_{H \setminus \{j\}}\}, \mathbf{vk}_{j, \overline{H}}^C, C_j, \eta_j, x_j^b, K_j^{\text{ct}}, \tau, K_j^{\text{EV}}],$$

we obfuscate

$$\text{EV}''_j[f, f_{H, \mathbf{x}_H^0}, \mathbf{vk}_{j, H \setminus \{j\}}^\rho \{\tilde{\rho}_{H \setminus \{j\}}\}, \mathbf{vk}_{j, \overline{H}}^\rho, \tilde{\rho}_j, \mathbf{vk}_{j, H \setminus \{j\}}^C \{C_{H \setminus \{j\}}\}, \mathbf{vk}_{j, \overline{H}}^C, C_j, \\ \eta_j, \tilde{x}_j^b, \tilde{K}_j^{\text{ct}}, \widetilde{\text{CTP}}_j, \tau, K_j^{\text{EV}}].$$

Here $\widetilde{\text{CTP}}_j$ is a copy of the public CTP obfuscation already contained in the view. The program EV''_j invokes this copy to obtain party j 's ciphertext instead of deriving the key and ciphertext directly using K_j^{ct} . It is specified in Figure 14.

Figure 14: Partial-decryption-simulating program with embedded CTP EV''_j

Hardwired:

- f , the circuit to be homomorphically evaluated;
- $f_{H, \mathbf{x}_H^0}$;
- the programmed reference-string and commitment verification keys for $H \setminus \{j\}$, and the ordinary keys for $\overline{H}$;
- $\tilde{\rho}_j$, C_j , and η_j ;
- $\widetilde{\text{CTP}}_j$, a copy of the public ciphertext-and-proof obfuscation;
- τ , the MTNIZK extraction trapdoor; and
- K_j^{EV} .

Input: ρ_{-j} , $\sigma_{j, -j}^\rho$, C_{-j} , $\sigma_{j, -j}^C$, $\tilde{\eta}_{-j}$, π_{-j}^η , ct_{-j} , and π_{-j}^{ct} .

Computation:

1. **Verify signatures:** For every $i \neq j$, check

$$\text{PSIG.Ver}(vk_{j,i}^\rho, \rho_i, \sigma_{j,i}^\rho) = 1 \quad \text{and} \quad \text{PSIG.Ver}(vk_{j,i}^C, C_i, \sigma_{j,i}^C) = 1.$$

If any check fails, output $\perp$.

2. **Verify MHE-string-share proofs:** Let $\tilde{\rho}$ be determined by $\tilde{\rho}_j$ and ρ_{-j} . For every $i \neq j$, check

$$\text{MTNIZK.V}(\tilde{\rho}, i, \tilde{\eta}_i, \pi_i^\eta) = 1.$$

If any check fails, output $\perp$.

3. **Aggregate the MHE string:** Compute

$$\eta := \eta_j \bigoplus_{i \in [m] \setminus \{j\}} \tilde{\eta}_i.$$

Figure 14 (continued)

4. **Verify ciphertext proofs:** For every $i \neq j$, check

$$\text{MTNIZK.V}(\tilde{\rho}, i, (\eta, C_i, ct_i), \pi_i^{ct}) = 1.$$

If any check fails, output $\perp$.

5. **Recover party j 's ciphertext:** Invoke the embedded obfuscation:

$$(ct_j, \pi_j^{ct}) := \widetilde{\text{CTP}}_j(\rho_{-j}, \sigma_{j,-j}^\rho, \tilde{\eta}_{-j}, \pi_{-j}^\eta).$$

6. **Evaluate:** Let $ct := (ct_1, \dots, ct_m)$ and compute

$$ct^* := \text{MHE.Eval}_\eta(f, ct).$$

7. For every $i \neq j$, extract

$$\begin{aligned} (x_i, r_i, s_i^{\text{sek}}) &:= \text{MTNIZK.E}(\tilde{\rho}, \tau, i, (\eta, C_i, ct_i), \pi_i^{ct}), \\ \text{sek}_i &:= G_{\text{sek}}(s_i^{\text{sek}}). \end{aligned}$$

8. Let $\mathbf{x}_{\bar{H}} := (x_i)_{i \in \bar{H}}$ and compute

$$y := f_{H, \mathbf{x}_H^0}(\mathbf{x}_{\bar{H}}).$$

9. Derive

$$r_j^{\text{dec}} := \text{PPRF.Eval}(K_j^{\text{EV}}, (\rho_{-j}, C_{-j}, \tilde{\eta}_{-j}, ct_{-j})),$$

and output

$$d_j \leftarrow \text{MHE.PDec}_\eta(j, \text{sek}_{-j}, ct^*, y; r_j^{\text{dec}}).$$

The two programs are functionally equivalent. If an outer verification check fails, both output $\perp$. Otherwise, all checks performed by the embedded $\widetilde{\text{CTP}}_j$ pass. This obfuscation contains the same x_j^b and K_j^{ct} that were hardwired directly into EV'_j , and it is invoked on the same $(\rho_{-j}, \tilde{\eta}_{-j})$ values. It therefore produces the same ct_j ; its proof output is discarded. All remaining computations are identical. Indistinguishability obfuscation gives

$$\text{Hyb}_6^b \approx_c \text{Hyb}_7^b.$$

Hybrid 8: Simulating the Ciphertext Proof and Replacing the Encrypted Input. Replace both the public obfuscation $\widetilde{\text{CTP}}_j$ and the copy hardwired into EV''_j . Instead of obfuscating

$$\text{CTP}_j \left[\mathbf{vk}_{j, H \setminus \{j\}}^\rho \{ \tilde{\rho}_{H \setminus \{j\}} \}, \mathbf{vk}_{j, \bar{H}}^\rho, \tilde{\rho}_j, \eta_j, C_j, x_j^b, r_j, K_j^{\text{ct}} \right],$$

we obfuscate

$$\text{CTP}'_j \left[\mathbf{vk}_{j, H \setminus \{j\}}^\rho \{ \tilde{\rho}_{H \setminus \{j\}} \}, \mathbf{vk}_{j, \bar{H}}^\rho, \tilde{\rho}_j, \eta_j, C_j, \mathbf{x}_j^b, \mathbf{r}_j, \tau, K_j^{\text{ct}} \right].$$

The program CTP'_j simulates its MTNIZK proof using τ and encrypts 0^n instead of x_j^b . In particular, it contains neither x_j^b nor the commitment opening r_j . The program is specified in Figure 15.

Figure 15: Simulated ciphertext-and-proof program CTP'_j

Hardwired:

- the programmed and ordinary reference-string verification keys, together with $\tilde{\rho}_j$;
- η_j and C_j ;
- τ , the MTNIZK simulation trapdoor; and
- K_j^{ct} .

Input: ρ_{-j} , $\sigma_{j,-j}^\rho$, $\tilde{\eta}_{-j}$, and π_{-j}^η .

Computation:

1. **Verify signatures:** For every $i \neq j$, check

$$\text{PSIG.Ver}(vk_{j,i}^\rho, \rho_i, \sigma_{j,i}^\rho) = 1.$$

If any check fails, output $\perp$.

2. **Verify MHE-string-share proofs:** Let $\tilde{\rho}$ be determined by $\tilde{\rho}_j$ and ρ_{-j} . For every $i \neq j$, check

$$\text{MTNIZK.V}(\tilde{\rho}, i, \tilde{\eta}_i, \pi_i^\eta) = 1.$$

If any check fails, output $\perp$.

3. **Aggregate the MHE string:** Compute

$$\eta := \eta_j \bigoplus_{i \in [m] \setminus \{j\}} \tilde{\eta}_i.$$

4. **Derive the encryption key and prover randomness:** Compute

$$s_j^{\text{sek}} \| r_j^{\text{ct}} := \text{PPRF.Eval}(K_j^{\text{ct}}, (\rho_{-j}, \tilde{\eta}_{-j})), \quad \text{sek}_j := G_{\text{sek}}(s_j^{\text{sek}}).$$

5. **Encrypt and prove:** Compute

$$ct_j := \text{MHE.Enc}_\eta(\text{sek}_j, 0^n)$$

and then, using tag j , simulate a proof for the statement (η, C_j, ct_j) :

$$\pi_j^{\text{ct}} := \text{MTNIZK.P}(\tilde{\rho}, \tau, j, (\eta, C_j, ct_j); r_j^{\text{ct}}).$$

Output $(ct_j, \pi_j^{\text{ct}})$.

We show that this change is indistinguishable using the Nondeterministic pIO Lemma 2. Consider the NP relation

$$R_{\text{CTP}}[\mathbf{vk}_{j,H \setminus \{j\}}^\rho \{\tilde{\rho}_{H \setminus \{j\}}\}, \mathbf{vk}_{j,\bar{H}}^\rho, \tilde{\rho}_j]((\rho_{-j}, \tilde{\eta}_{-j}), (\sigma_{j,-j}^\rho, \pi_{-j}^\eta)).$$

Its instance is $(\rho_{-j}, \tilde{\eta}_{-j})$ and its witness is $(\sigma_{j,-j}^\rho, \pi_{-j}^\eta)$. The relation accepts if and only if all signature and MHE-string-share proof checks performed by CTP_j and CTP'_j accept. Let $L_{\text{CTP},\lambda}$

denote its language.

We now show the required pointwise indistinguishability toward applying the Nondeterministic pIO Lemma 2. By the assumed unique verification, every instance $(\rho_{-j}, \tilde{\eta}_{-j}) \in L_{\text{CTP}, \lambda}$ satisfies

$$\rho_{H \setminus \{j\}} = \tilde{\rho}_{H \setminus \{j\}}.$$

Consequently, the reference string used by every accepted computation has its coordinates in H programmed at tag j . There are at most $2^{(m-1)\lambda}$ possible vectors ρ_{-j} . Moreover, extraction from every accepting MHE-string-share proof shows that each accepted $\tilde{\eta}_i$ is generated from an ℓ_η -bit seed, where $\ell_\eta = \lambda$. Thus,

$$|L_{\text{CTP}, \lambda}| \leq 2^{2(m-1)\lambda}.$$

For every instance $(\rho_{-j}, \tilde{\eta}_{-j}) \in L_{\text{CTP}, \lambda}$, first consider the intermediate program that encrypts x_j^b , as CTP_j does, but simulates the ciphertext proof. The statement proved by CTP_j is true: C_j commits to x_j^b , and ct_j is formed as prescribed by R_{ct} . Conditioned on the common generated ciphertext, ℓ -zero knowledge gives

$$\begin{aligned} & \text{MTNIZK.P}(\tilde{\rho}, j, (\eta, C_j, ct_j), (x_j^b, r_j, s_j^{\text{sek}})) \\ & \approx_{2^{-\ell} \cdot \text{negl}(\lambda)} \text{MTNIZK.P}(\tilde{\rho}, \tau, j, (\eta, C_j, ct_j)). \end{aligned}$$

Thus, this first transition has pointwise distinguishing advantage at most $2^{-\ell} \cdot \text{negl}(\lambda)$. Since $\ell \geq 2m\lambda$, this is at most

$$\frac{\text{negl}(\lambda)}{|L_{\text{CTP}, \lambda}|}.$$

It remains to replace the ciphertext generated by the intermediate program. Namely, we show that the distributions

$$\left(\eta, \text{MHE.Enc}_\eta(G_{\text{sek}}(s_j^{\text{sek}}), x_j^b) \right) \quad \text{and} \quad \left(\eta, \text{MHE.Enc}_\eta(G_{\text{sek}}(s_j^{\text{sek}}), 0^n) \right)$$

are computationally indistinguishable.

By subexponential security of G_{sek} we can replace $G_{\text{sek}}(s_j^{\text{sek}})$ by a uniform MHE key contributing to the pointwise advantage at most

$$2^{-\ell_{\text{sek}}^\varepsilon} \leq 2^{-2m\lambda} \leq \frac{2^{-2\lambda}}{|L_{\text{CTP}, \lambda}|} \leq \frac{\text{negl}(\lambda)}{|L_{\text{CTP}, \lambda}|}.$$

For the fixed instance $(\rho_{-j}, \tilde{\eta}_{-j})$, the vector $\tilde{\eta}_{-j}$ is fixed. Since η_j is uniform, the aggregate string $\eta = \eta_j \oplus_{i \in [m] \setminus \{j\}} \tilde{\eta}_i$ is uniform. Semantic security of MHE then allows replacing the encryption of x_j^b by an encryption of 0^n contributing to the pointwise advantage at most

$$2^{-\tilde{\lambda}^\varepsilon} \leq 2^{-m(n+3\lambda+\ell_{\text{sek}})} \leq \frac{\text{negl}(\lambda)}{|L_{\text{CTP}, \lambda}|}.$$

Combining the zero-knowledge, PRG, and MHE transitions, the pointwise distinguishing advantage between CTP_j and CTP'_j is at most $\text{negl}(\lambda)/|L_{\text{CTP}, \lambda}|$. By the choice of $\tilde{\lambda}$, the advantages of iO and PPRF are at most

$$2^{-\tilde{\lambda}^\varepsilon} < \frac{\text{negl}(\lambda)}{|L_{\text{CTP}, \lambda}|}.$$

Note that K_j^{ct} was removed in Hybrid 7. Hence we can apply the Nondeterministic pIO Lemma 2, deducing that

$$\widetilde{\text{CTP}}_j \approx_c \widetilde{\text{CTP}}'_j.$$

Thus,

$$\text{Hyb}_7^b \approx_c \text{Hyb}_8^b.$$

Hybrid 9: Replacing Party j 's Commitment. Replace party j 's commitment with a commitment to zero:

$$C_j := \text{Com}(x_j^b; r_j) \quad \text{by} \quad \tilde{C}_j \leftarrow \text{Com}(0^n).$$

The opening r_j of the commitment was removed in Hybrid 8 and hence by computational hiding of the commitment

$$\text{Hyb}_8^b \approx_c \text{Hyb}_9^b.$$

Conclusion. The distribution Hyb_9^b is independent of b . Party j 's public commitment and generated ciphertext concern 0^n ; its MHE-string-share proof and ciphertext proof are simulated; and its evaluation program depends only on the common residual function f_{H, x_H^0} . All other honest parties have the same input in the two experiments. Thus,

$$\text{Hyb}_9^0 \equiv \text{Hyb}_9^1.$$

Combining the two hybrid sequences proves neighboring-input $\mathcal{H}_{\text{maj}}$ -robustness.

Padding. All obfuscated programs in the construction of Figure 10 are padded to the size of the largest corresponding program appearing in the hybrids above. $\square$

Proposition 2 and Lemma 1 imply that the construction is indistinguishability $\mathcal{H}_{\text{maj}}$ -robust.

AI Usage

We have relied on OpenAI's Prism (5.6 Sol) for (a) editing, formatting, and figure generation; (b) reviewing definitions and proofs; (c) generating the example in Remark 4, Appendix A.

Acknowledgments

This research was partially supported by the Advanced Research and Invention Agency (ARIA). We thank Omer Paneth and Idan Halawi for discussions during the early stages of this work. We also thank OpenAI's Prism for useful assistance with writing this paper.

References

- [ABT18] Benny Applebaum, Zvika Brakerski, and Rotem Tsabary. Perfect secure computation in two rounds. In *Theory of Cryptography Conference–TCC 2018, Part I*, volume 11239 of Lecture Notes in Computer Science, pages 152–174. Springer, 2018.
- [AJJM20] Prabhanjan Ananth, Abhishek Jain, Zhengzhong Jin, and Giulio Malavolta. Multi-key fully-homomorphic encryption in the plain model. In *Theory of Cryptography Conference–TCC 2020, Part I*, volume 12550 of Lecture Notes in Computer Science, pages 28–57. Springer, 2020.
- [AIK06] Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. Cryptography in NC^0 . *SIAM Journal on Computing*, 36(4):845–888, 2006.
- [AS15] Gilad Asharov and Gil Segev. Limits on the power of indistinguishability obfuscation and functional encryption. In *Proceedings of the 56th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 191–209, 2015.

- [BD19] Nir Bitansky and Akshay Degwekar. On the complexity of collision-resistant hash functions: New and old black-box separations. In *Theory of Cryptography Conference–TCC 2019, Part I*, volume 11891 of Lecture Notes in Computer Science, pages 422–450. Springer, 2019.
- [BDV17] Nir Bitansky, Akshay Degwekar, and Vinod Vaikuntanathan. Structure vs. hardness through the obfuscation lens. In *Advances in Cryptology–CRYPTO 2017, Part I*, volume 10401 of Lecture Notes in Computer Science, pages 696–723. Springer, 2017.
- [BGI+14] Amos Beimel, Ariel Gabizon, Yuval Ishai, Eyal Kushilevitz, Sigurd Meldgaard, and Anat Paskin-Cherniavsky. Non-interactive secure multiparty computation. In *Advances in Cryptology–CRYPTO 2014, Part II*, volume 8617 of Lecture Notes in Computer Science, pages 387–404. Springer, 2014.
- [FKN94] Uriel Feige, Joe Kilian, and Moni Naor. A minimal model for secure computation. In *Proceedings of the 26th Annual ACM Symposium on Theory of Computing (STOC)*, pages 554–563, 1994.
- [GGG+14] Shafi Goldwasser, S. Dov Gordon, Vipul Goyal, Abhishek Jain, Jonathan Katz, Feng-Hao Liu, Amit Sahai, Elaine Shi, and Hong-Sheng Zhou. Multi-input functional encryption. In *Advances in Cryptology–EUROCRYPT 2014*, volume 8441 of Lecture Notes in Computer Science, pages 578–602. Springer, 2014.
- [HIJ+16] Shai Halevi, Yuval Ishai, Abhishek Jain, Eyal Kushilevitz, and Tal Rabin. Secure multiparty computation with general interaction patterns. In *Proceedings of the 7th Innovations in Theoretical Computer Science Conference (ITCS)*, pages 157–168, 2016.
- [IKOS06] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Cryptography from anonymity. In *Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 239–248, 2006.
- [IK00] Yuval Ishai and Eyal Kushilevitz. Randomizing polynomials: A new representation with applications to round-efficient secure computation. In *Proceedings of the 41st Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 294–304, 2000.
- [BEG25] Nir Bitansky, Saroja Erabelli, and Rachit Garg. Additive randomized encodings from public key encryption. In *Advances in Cryptology–CRYPTO 2025*, pages 241–260, 2025.
- [BBHR18] Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. Scalable, transparent, and post-quantum secure computational integrity. Cryptology ePrint Archive, Report 2018/046, 2018.
- [BEGI26] Nir Bitansky, Saroja Erabelli, Rachit Garg, and Yuval Ishai. Shuffling is universal: Statistical additive randomized encodings for all functions. In *Proceedings of the 58th Annual ACM Symposium on Theory of Computing (STOC)*, 2026.
- [BF24] Nir Bitansky and Sapir Freizeit. Robust additive randomized encodings from iO and pseudo-non-linear codes. In *Advances in Cryptology–CRYPTO 2024*, pages 109–135, 2024.
- [BOV03] Boaz Barak, Shien Jin Ong, and Salil Vadhan. Derandomization in cryptography. In *Advances in Cryptology–CRYPTO 2003*, volume 2729 of Lecture Notes in Computer Science, pages 299–315. Springer, 2003.

- [BP15] Nir Bitansky and Omer Paneth. ZAPs and non-interactive witness indistinguishability from indistinguishability obfuscation. In *Theory of Cryptography Conference–TCC 2015, Part II*, volume 9015 of Lecture Notes in Computer Science, pages 401–427. Springer, 2015.
- [BPW16] Nir Bitansky, Omer Paneth, and Daniel Wichs. Perfect structure on the edge of chaos: Trapdoor permutations from indistinguishability obfuscation. In *Theory of Cryptography Conference–TCC 2016-A, Part I*, volume 9562 of Lecture Notes in Computer Science, pages 474–502. Springer, 2016.
- [DHRW16] Yevgeniy Dodis, Shai Halevi, Ron D. Rothblum, and Daniel Wichs. Spooky encryption and its applications. In *Advances in Cryptology–CRYPTO 2016*, pages 93–122, 2016.
- [DN00] Cynthia Dwork and Moni Naor. Zaps and their applications. In *Proceedings of the 41st Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 283–293, 2000.
- [BGI+01] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. In *Advances in Cryptology–CRYPTO 2001*, volume 2139 of Lecture Notes in Computer Science, pages 1–18. Springer, 2001.
- [CLTV15] Ran Canetti, Huijia Lin, Stefano Tessaro, and Vinod Vaikuntanathan. Obfuscation of probabilistic circuits and applications. In *Theory of Cryptography Conference–TCC 2015, Part II*, volume 9015 of Lecture Notes in Computer Science, pages 468–497. Springer, 2015.
- [CM15] Michael Clear and Ciarán McGoldrick. Multi-identity and multi-key leveled FHE from learning with errors. In *Advances in Cryptology–CRYPTO 2015, Part II*, volume 9216 of Lecture Notes in Computer Science, pages 630–656. Springer, 2015.
- [GO07] Jens Groth and Rafail Ostrovsky. Cryptography in the multi-string model. In *Advances in Cryptology–CRYPTO 2007*, volume 4622 of Lecture Notes in Computer Science, pages 323–341. Springer, 2007.
- [GOS06] Jens Groth, Rafail Ostrovsky, and Amit Sahai. Non-interactive zaps and new techniques for NIZK. In *Advances in Cryptology–CRYPTO 2006*, volume 4117 of Lecture Notes in Computer Science, pages 97–111. Springer, 2006.
- [Gro06] Jens Groth. Simulation-sound NIZK proofs for a practical language and constant size group signatures. In *Advances in Cryptology–ASIACRYPT 2006*, volume 4284 of Lecture Notes in Computer Science, pages 444–459. Springer, 2006.
- [GGSW13] Sanjam Garg, Craig Gentry, Amit Sahai, and Brent Waters. Witness encryption and its applications. In *Proceedings of the 45th Annual ACM Symposium on Theory of Computing (STOC)*, pages 467–476, 2013.
- [GM11] Vipul Goyal and Hemanta K. Maji. Stateless cryptographic protocols. In *Proceedings of the 52nd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 678–687, 2011.

- [GS09] Vipul Goyal and Amit Sahai. Resettably secure computation. In *Advances in Cryptology–EUROCRYPT 2009*, volume 5479 of Lecture Notes in Computer Science, pages 54–71. Springer, 2009.
- [HIKR23] Shai Halevi, Yuval Ishai, Eyal Kushilevitz, and Tal Rabin. Additive randomized encodings and their applications. In *Advances in Cryptology–CRYPTO 2023*, pages 203–235, 2023.
- [HIJ+17] Shai Halevi, Yuval Ishai, Abhishek Jain, Ilan Komargodski, Amit Sahai, and Eylon Yogev. Non-interactive multiparty computation without correlated randomness. In *Advances in Cryptology–ASIACRYPT 2017, Part III*, volume 10626 of Lecture Notes in Computer Science, pages 181–211. Springer, 2017.
- [JLS21] Aayush Jain, Huijia Lin, and Amit Sahai. Indistinguishability obfuscation from well-founded assumptions. In *Proceedings of the 53rd Annual ACM Symposium on Theory of Computing (STOC)*, pages 60–73, 2021.
- [JLS22] Aayush Jain, Huijia Lin, and Amit Sahai. Indistinguishability obfuscation from LPN over $\mathbb{F}_p$, DLIN, and PRGs in NC^0 . In *Advances in Cryptology–EUROCRYPT 2022, Part I*, volume 13275 of Lecture Notes in Computer Science, pages 670–699. Springer, 2022.
- [HLP11] Shai Halevi, Yehuda Lindell, and Benny Pinkas. Secure computation on the web: Computing without simultaneous interaction. In *Advances in Cryptology–CRYPTO 2011*, volume 6841 of Lecture Notes in Computer Science, pages 132–150. Springer, 2011.
- [KLW15] Venkata Koppula, Allison Bishop Lewko, and Brent Waters. Indistinguishability obfuscation for Turing machines with unbounded memory. In *Proceedings of the 47th Annual ACM Symposium on Theory of Computing (STOC)*, pages 419–428, 2015.
- [Lau83] Clemens Lautemann. BPP and the polynomial hierarchy. *Information Processing Letters*, 17(4):215–217, 1983.
- [MW16] Pratyay Mukherjee and Daniel Wichs. Two round multiparty computation via multi-key FHE. In *Advances in Cryptology–EUROCRYPT 2016, Part II*, volume 9666 of Lecture Notes in Computer Science, pages 735–763. Springer, 2016.
- [Nao91] Moni Naor. Bit commitment using pseudorandomness. *Journal of Cryptology*, 4(2):151–158, 1991.
- [Sah99] Amit Sahai. Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In *Proceedings of the 40th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 543–553, 1999.
- [BGI14] Elette Boyle, Shafi Goldwasser, and Ioana Ivan. Functional signatures and pseudorandom functions. In *Public-Key Cryptography–PKC 2014*, pages 501–519, 2014.
- [BW13] Dan Boneh and Brent Waters. Constrained pseudorandom functions and their applications. In *Advances in Cryptology–ASIACRYPT 2013*, pages 280–300, 2013.

- [KPTZ13] Aggelos Kiayias, Stavros Papadopoulos, Nikos Triandopoulos, and Thomas Zacharias. Delegatable pseudorandom functions and applications. In *Proceedings of the 20th ACM Conference on Computer and Communications Security (CCS)*, pages 669–684, 2013.
- [WSAZ26] Yu Wei, Jaspal Singh, Adya Agrawal, and Vassilis Zikas. Balanced additive randomized encodings for shuffle differential privacy. Cryptology ePrint Archive, Report 2026/1239, 2026.

A From Neighboring-Input $\mathcal{H}$ -Robustness to Full $\mathcal{H}$ -Robustness

In this section we prove:

Lemma 1 (restated). *If every function in $\mathsf{P/poly}$ (respectively NC^1) admits a neighboring-input $\mathcal{H}$ -robust additive randomized encoding, then every function in $\mathsf{P/poly}$ (respectively NC^1) admits an $\mathcal{H}$ -robust additive randomized encoding (in the standard sense).*

Proof of Lemma 1. Let $\mathcal{X} := \{0, 1\}^n$ and $\mathcal{Y} := \{0, 1\}^*$ be the input and output domains of f , respectively. We define an augmented function $\hat{f}$ that allows a party to provide its own input or to delegate this choice to another party. Assume without loss of generality that $\mathcal{X}$, $[m]$, and $\{\perp\}$ are pairwise disjoint, and let

$$\hat{f}: ((\mathcal{X} \cup [m] \cup \{\perp\})^m)^m \rightarrow \mathcal{Y} \cup \{\perp\}.$$

The input of party i is a vector $\mathbf{y}_i = (y_{i,1}, \dots, y_{i,m})$. To determine the input of party i , the function first examines the diagonal entry $y_{i,i}$. If this entry is a value in $\mathcal{X}$ (or $\perp$), it is used directly. If $y_{i,i} = j \in [m]$, then party i delegates its input to party j , and the value $y_{j,i}$ is used. Formally, the resolved input of party i is

$$\hat{y}_i := \begin{cases} y_{i,i}, & y_{i,i} \in \mathcal{X} \cup \{\perp\}, \\ y_{j,i}, & y_{i,i} = j \in [m]. \end{cases}$$

If all resolved inputs belong to $\mathcal{X}$, the function outputs $f(\hat{y}_1, \dots, \hat{y}_m)$; otherwise, it outputs $\perp$.

Resolving all inputs can be implemented by a circuit of size $\tilde{O}(m^2n)$ and depth $O(\log(mn))$. Consequently, $\hat{f}$ belongs to $\mathsf{P/poly}$ (respectively, NC^1) whenever f does.

Instantiate the neighboring-input $\mathcal{H}$ -robust RARE with $\hat{f}$. In an ordinary execution, to encode its input x_i , party i forms an m -dimensional vector containing x_i in coordinate i and $\perp$ in all other coordinates, and encodes this vector using the RARE for $\hat{f}$. The decoder is unchanged.

Correctness. In an ordinary execution, every party provides its own input directly, so correctness for f follows immediately from the definition of $\hat{f}$.

Security. We prove robustness. Fix $H = \{h_1, \dots, h_k\} \in \mathcal{H}$ and input vectors $\mathbf{x}_H^0 = (x_i^0)_{i \in H}$ and $\mathbf{x}_H^1 = (x_i^1)_{i \in H}$ satisfying

$$f_{H, \mathbf{x}_H^0} \equiv f_{H, \mathbf{x}_H^1}.$$

For each $b \in \{0, 1\}$, we define augmented input vectors $\mathbf{z}_H^{b,0}, \dots, \mathbf{z}_H^{b,k}$ as follows. Initially, in $\mathbf{z}_H^{b,0}$, every honest party $i \in H$ provides x_i^b directly: its vector contains x_i^b in coordinate i and $\perp$ elsewhere.

To obtain $\mathbf{z}_H^{b,1}$, we change only the input of h_1 . Party h_1 now places x_i^b in coordinate i for every $i \in H$, and $\perp$ in every coordinate outside H . Thus, h_1 holds all the honest inputs, although the other honest parties still provide their inputs directly. Next, for $p = 2, \dots, k$, we obtain $\mathbf{z}_H^{b,p}$ from

$z_H^{b,p-1}$ by changing only the input of h_p : party h_p replaces its value $x_{h_p}^b$ with the index h_1 , thereby delegating its input to h_1 . Consequently, in $z_H^{b,k}$, every honest party other than h_1 delegates its input to h_1 . Figure 16 illustrates these changes under the simplifying assumption $H = [k]$.

Consider the hybrid sequence

$$z_H^{0,0}, z_H^{0,1}, \dots, z_H^{0,k}, z_H^{1,k}, z_H^{1,k-1}, \dots, z_H^{1,0}.$$

Every adjacent pair differs in the augmented input of one honest party. For a fixed b , all these hybrids resolve to the same honest input vector $\mathbf{x}_H^b$: the only change is whether an input is supplied directly or obtained from h_1 . The entries of every honest vector outside H remain $\perp$, so the behavior on the inputs of $\bar{H}$ is also unchanged. Hence adjacent hybrids on either side of the middle pair induce identical residual functions for $\hat{f}$.

The middle pair also differs only in the input of h_1 , because the inputs of $h_2, \dots, h_k$ are delegation vectors independent of b . Its residual functions are identical by the assumption $f_{H, \mathbf{x}_H^0} \equiv f_{H, \mathbf{x}_H^1}$: the honest coordinates resolve to $\mathbf{x}_H^0$ and $\mathbf{x}_H^1$, respectively, while the treatment of every coordinate outside H is unchanged.

Neighboring-input $\mathcal{H}$ -robustness therefore makes the RARE distributions for every adjacent pair computationally indistinguishable. By a hybrid argument, the distributions corresponding to $z_H^{0,0}$ and $z_H^{1,0}$ are computationally indistinguishable. These are the distributions of the compiled RARE for f on $\mathbf{x}_H^0$ and $\mathbf{x}_H^1$, respectively. Hence the compiled RARE satisfies indistinguishability $\mathcal{H}$ -robustness. $\square$

$$\begin{aligned} z_{[k]}^{b,0} &= \begin{pmatrix} x_1^b & \perp & \cdots & \perp \\ \perp & x_2^b & \ddots & \vdots \\ \vdots & \ddots & \ddots & \perp \\ \perp & \cdots & \perp & x_k^b \end{pmatrix} \xrightarrow{\text{party 1 gathers the honest inputs}} z_{[k]}^{b,1} = \begin{pmatrix} x_1^b & x_2^b & \cdots & x_k^b \\ \perp & x_2^b & \ddots & \vdots \\ \vdots & \ddots & \ddots & \perp \\ \perp & \cdots & \perp & x_k^b \end{pmatrix} \\ z_{[k]}^{b,1} &\xrightarrow{\text{parties } 2, \dots, k \text{ delegate to party 1, one at a time}} z_{[k]}^{b,k} = \begin{pmatrix} x_1^b & x_2^b & \cdots & x_k^b \\ \perp & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \perp \\ \perp & \cdots & \perp & 1 \end{pmatrix}. \end{aligned}$$

Figure 16: The input-delegation hybrids for $H = [k]$. Row i is the input vector of party i , and column j contains the value that party i can supply for party j . Only the $H \times H$ block is shown; all coordinates outside H remain $\perp$.

Remark 4 (The gap in the proof of [HIJ+17]). Halevi et al. [HIJ+17] define an augmented function over $((\mathcal{X} \cup \{\perp\})^m)^m$. If a unique party submits a complete tuple $(x_1, \dots, x_m)$ containing no $\perp$ entries, the function outputs $f(x_1, \dots, x_m)$; otherwise, it interprets the parties' vectors as singleton inputs and applies f to the extracted values. The proposed hybrid strategy consolidates the honest inputs into the vector of one honest party, and gradually erases all the others. When $H \neq [m]$, however, the designated honest party cannot supply the corrupted parties' inputs. Placing $\perp$ in those coordinates prevents the complete-tuple branch from applying, while filling them with fixed values does not preserve the residual function over the inputs of the corrupted parties. For example, let $m = 3$, $H = \{1, 2\}$, and $f(x_1, x_2, x_3) = x_3$. Consider the adjacent honest inputs

$$((1, \perp, \perp), (1, \perp, \perp)) \quad \text{and} \quad ((1, 1, c), (1, \perp, \perp)),$$

where $c \in \{0, 1\}$ is the fixed value used to complete party 1's tuple. If corrupted party 3 submits $(1 - c, \perp, \perp)$, the first hybrid takes the singleton-input branch and outputs $1 - c$, whereas the second takes the complete-tuple branch and outputs c . Hence these hybrids induce different residual functions and are distinguishable by correctness. Our augmented function avoids this problem by allowing coordinate-wise delegation: one designated honest party supplies only the honest coordinates, while every corrupted party retains control of its own input.